AppGameKit STUDIO User Guide

Welcome to the AppGameKit Studio User Guide. The purpose of the guide is to introduce you to the many tools and features so you can quickly get up and running with AppGameKit studio.

AppGameKit Studio is a fully featured game development tool set. We've re-imagined the game development user interface with an all-in-one work space. You can now quickly drag & drop assets to visualise your scenes, code with AppGameKit Script, easily browse app media, run live debugging sessions, access online help and lots more.

AppGameKit Studio allows you, in a few easy clicks, to quickly publish your games to Steam, iOS, Android and a host of other places! Get your apps and games in front of millions of potential players and make yours the next big hit.

New Vulkan Rendering Engine!

This first release of AppGameKit Studio features both OpenGL and Vulkan rendering engines.

The OpenGL engine has been improved, faster, fully complete and works cross platform.

The new Vulkan engine is currently in BETA. For now, the Vulkan engine is only available on the Windows platform.

Choosing which rendering engine to run

Since the release of AppGameKit Studio 2019.08.05 it’s been possible to choose which rendering engine Studio will use when you run your app. This will be helpful if you are having issues running Vulkan on your PC. The command to use is #renderer

So, if you add this code to the start of your project:

```
#renderer "Basic"     // This will choose the OpenGL renderer

#renderer "Advanced" // This will choose Vulkan

#renderer "Prefer Best" // This will use the default system which will
first try to use Vulkan and if that's not possible on the hardware it
will fall back to OpenGL
```

When you run your app on Windows, AppGameKit Studio will check the hardware it's running on and choose to run the most appropriate rendering option. Older devices may only be able to run OpenGL so they will default to OpenGL while newer hardware will make use of the faster Vulkan engine if supported. It's totally transparent so you don't need to worry about choosing which engine to use. Even shaders are auto handled by AppGameKit Studio. Shaders start in GLSL V110 format. If the renderer is OpenGL then they are used as is in GLSL format. If the renderer is Vulkan then AppGameKit Studio converts them into SPIRV format.

Vulkan Updates Coming Soon!

We’re still working on the Vulkan engine and we hope that over the coming weeks we will release updates to support the other platforms in this order of release importance:

- Mac
- iOS
- Linux
- Android

If you experience any problems running your apps on OpenGL or Vulkan hardware then please report your issues to our team here: AppGameKit Studio Issues

CONTENTS

QUICK START GUIDE

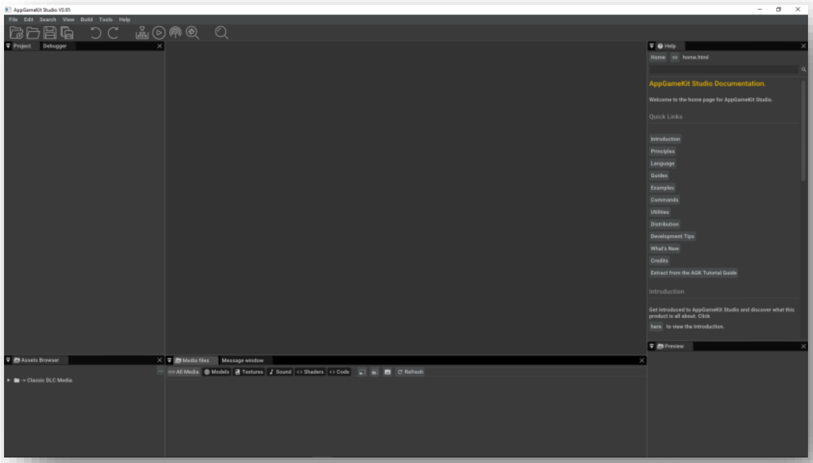
There's a lot to learn about AppGameKit Studio but we know you're itching to get creating and so we've created this quick start guide so you can see some immediate results. Coding is at the heart of AppGameKit Studio, it gives you total control over what you can create and doesn't restrict or box you in. We're going to use a mode called Sandbox to quickly get something on screen moving.

This is the small program you're going to type in. It will load in a backdrop image and a box sprite and then move the box sprite left to right and then back right to left. It might look complex at first but taken line by line it will all make sense and soon you will be reading and understanding AppGameKit Studio code:

```
SetVirtualResolution (320,480)
image1 = LoadImage("background5.jpg"  )
Sprite1 = CreateSprite ( image1 )
image2 = LoadImage ( "blue.png" )
sprite2 = CreateSprite ( image2 )
direction = 4
do
    SetSpritePosition ( sprite2, x, 240 )
    x = x + direction
       if x > 270 then direction = -4
       if x < 0 then direction = 4
    sync ( )
loop
```

Sandbox Mode

With AppGameKit Studio loaded and waiting for you to use it, you should see the main user interface awaiting your commands:



AppGameKit Studio V0.95
File Edit Search View Build Tools Help
Project Debugger
Help
Home
home.html
Assets Browser
Classic DLC Media
Media files Message window
All Media Models Textures Sound Shaders Code Refresh
Preview

AppGameKit Studio Documentation.

Welcome to the home page for AppGameKit Studio.

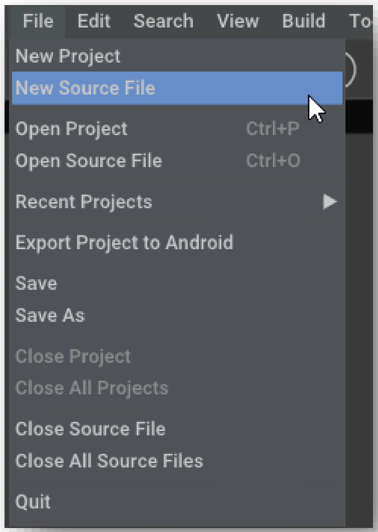
Quick Links

Introduction
Principles
Language
Guides
Examples
Commands
Utilities
Distribution
Development Tips
What's New
Credits
Extract from the AGK Tutorial Guide

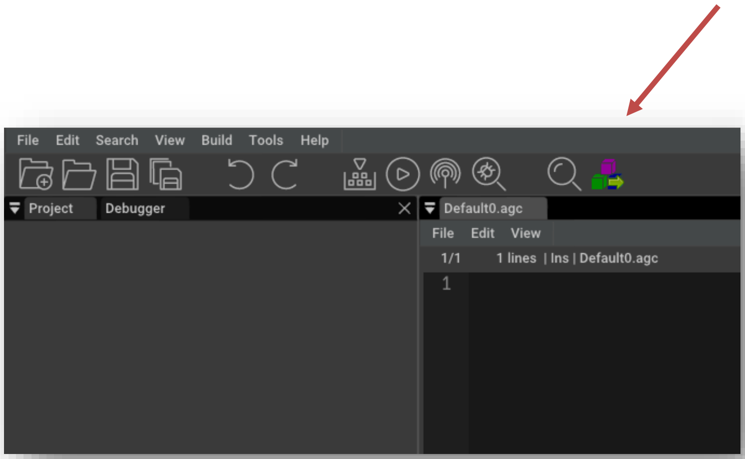
Introduction

Get introduced to AppGameKit Studio and discover what this product is all about. Click here to view the Introduction.

There's a special mode in AppGameKit Studio called Sandbox mode. To access this mode, you just need to select the *File* drop-down menu and then choose *New Source File*.



A new source file called *Default0.agc* will be created and the main IDE window will be opened up ready for you to type some programming code. The tool bar will also have a new sandbox icon added to it.



Setting a resolution

The first step is to set the resolution of our app. There are two modes in AppGameKit Studio – Virtual and Percentage resolutions. The default is percentage, but in our tutorial, we'll use virtual resolution. The resolution we're choosing is the same size as the background image we will be loading – 320 x 480 pixels.

Type the following text into the IDE and make sure it's exactly the same as this:

```
SetVirtualResolution (320,480)
```

You can now run this one-line program by clicking on the Sandbox icon. Watch carefully because the window will appear and then the program will end and it will be all over.

Loading images

The next section of the code loads in two image files and then makes two sprites that will use these images. Type in the following lines so that the full program code now looks like this:

```
SetVirtualResolution (320,480)
image1 = LoadImage("background5.jpg" )
Sprite1 = CreateSprite ( image1 )
image2 = LoadImage ( "blue.png" )
sprite2 = CreateSprite ( image2 )
```

The line "`image1 = LoadImage("background5.jpg" )`" will instruct AppGameKit Studio to load in the image file *background.jpg* and create an ID number for it. This ID number is stored in the variable called *image1*.

The next line "`Sprite1 = CreateSprite ( image1 )`", creates a Sprite called *Sprite1* and assigns the image that we just loaded to the newly created sprite.

The next two lines do the same with another image "`blue.png`", making a Sprite called *Sprite2.*

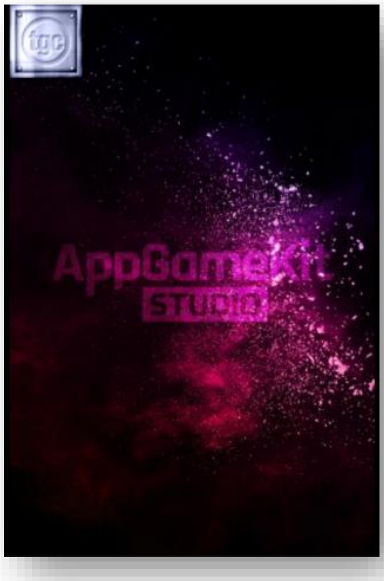
Do…Loop with a Sync!

If we were to run the above code, so far we would not see anything. This is because all the lines would be run through so quickly and the app would be finished and it would end. So, to help you see what you have done so far, you just need to add a small loop to the end of the program. A *Do..Loop* will keep the code between these two commands looping around and around forever. We add a *Sync()* command in between to ensure we wait for the screen to be drawn each frame.

Make sure your program looks like this and run it again by pressing the sandbox icon:

```
SetVirtualResolution (320,480)
image1 = LoadImage("background5.jpg" )
Sprite1 = CreateSprite ( image1 )
image2 = LoadImage ( "blue.png" )
sprite2 = CreateSprite ( image2 )
Do
   Sync()
Loop
```

If you've typed in every character properly then you will see this on screen:



The backdrop *Sprite1* is displayed and the smaller sprite is shown in the top left.

Now let's add some movement to the smaller sprite. We'll aim to move it from the left to the right and when it reaches the right we'll move back to the left. To achieve this, we will use a variable that we'll call "direction". We'll set "direction" to equal 4 and change it to -4 when the *Sprite2* reaches the right-hand side of the screen. Let's look at the code to achieve this:

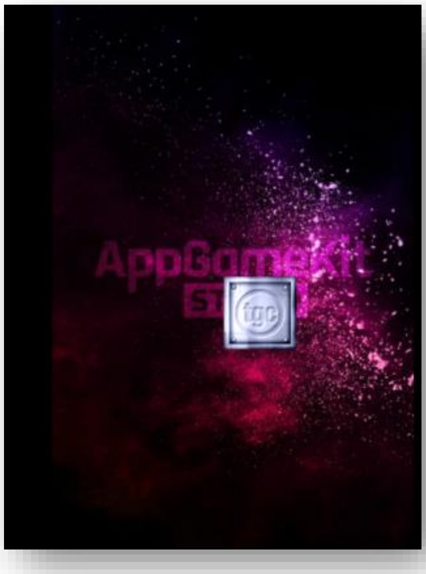
```
direction = 4
do
    SetSpritePosition ( sprite2, x, 240 )
    x = x + direction
       if x > 270 then direction = -4
       if x < 0 then direction = 4
    sync ( )
loop
```

Let’s break this down:

`direction = 4`	Sets an integer variable called “direction” to equal the value of 4.
`do`	This marks to the start of a do…loop, all code between the do and loop commands will keep repeating over and over.
`SetSpritePosition ( sprite2, x, 240 )`	This will set the sprite’s position of “sprite2”. It will use the value in variable x for the x axis position and a value of 240 for the y axis position. On first run, x will default to zero.
`x = x + direction`	The value x is now changed. It will be increased by the value of direction. So, if x = 0 and direction = 4, then x will now equal 4. The next time around the loop it will equal 8 and so on.
`if x > 270 then direction = -4`	This checks to see if the x value is now greater than the value of 270. If it has then the variable direction is set to -4. Once direction has a negative value the line above will start reducing the value of x (moving the sprite left).
`if x < 0 then direction = 4`	Check to see if the x value is less than zero. If it is, set direction variable to equal positive 4 (move right.
`sync ( )`	Wait until the screen has been drawn for a frame (for smooth movement in time with the frame rate).
`loop`	Jump back to the do command above and continue the program from there.

As a recap, this is the code you should have typed out;

```
SetVirtualResolution (320,480)
image1 = LoadImage("background5.jpg" )
Sprite1 = CreateSprite ( image1 )
image2 = LoadImage ( "blue.png" )
sprite2 = CreateSprite ( image2 )
direction = 4
do
    SetSpritePosition ( sprite2, x, 240 )
    x = x + direction
       if x > 270 then direction = -4
       if x < 0 then direction = 4
    sync ( )
loop
```

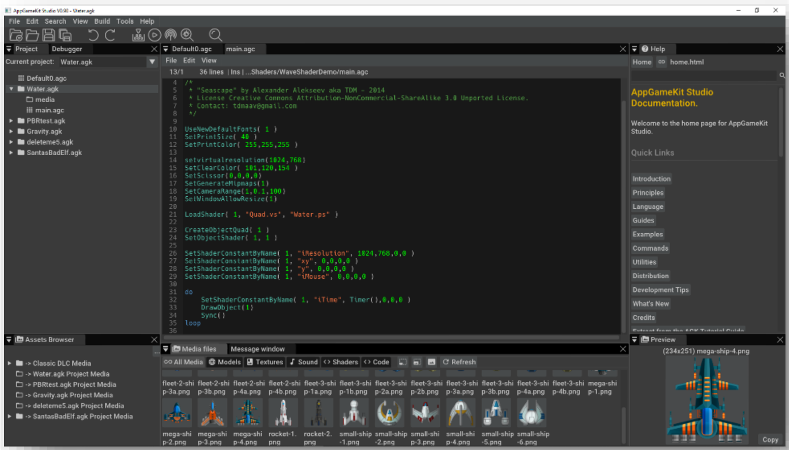


If you typed it in correctly then the image on the right is what you should see running in sandbox mode.

THE MAIN USER INTERFACE

AppGameKit Studio’s UI integrates the coding IDE with other important tools such as the projects list, scene manager, asset browser, help and more.

This is a typical screen showing how the UI can look like when you’re developing a project:



```
/*
 * "Seascape" by Alexander Alekseev aka TDM - 2014
 * License Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.
 * Contact: tdmaav@gmail.com
 */

UseNewDefaultFonts( 1 )
SetPrintSize( 40 )
SetPrintColor( 255,255,255 )

setvirtualresolution(1024,768)
SetClearColor( 101,120,154 )
SetScissor(0,0,0,0)
SetGenerateMipmaps(1)
SetCameraRange(1,0.1,100)
SetWindowAllowResize(1)

LoadShader( 1, "Quad.vs", "Water.ps" )

CreateObjectQuad( 1 )
SetObjectShader( 1, 1 )

SetShaderConstantByName( 1, "iResolution", 1024,768,0,0 )
SetShaderConstantByName( 1, "xy", 0,0,0,0 )
SetShaderConstantByName( 1, "y", 0,0,0,0 )
SetShaderConstantByName( 1, "iMouse", 0,0,0,0 )

do
    SetShaderConstantByName( 1, "iTime", Timer(),0,0,0 )
    DrawObject(1)
    Sync()
loop
```

Help on each area of the UI is covered in much more detail within this user guide.

The main User Interface

At the top of the screen are the drop-down menus and the quick access tool bar.

On the left side are the project and debugger windows.

The centre is where the main IDE is located and it can also show the Scene Editor.

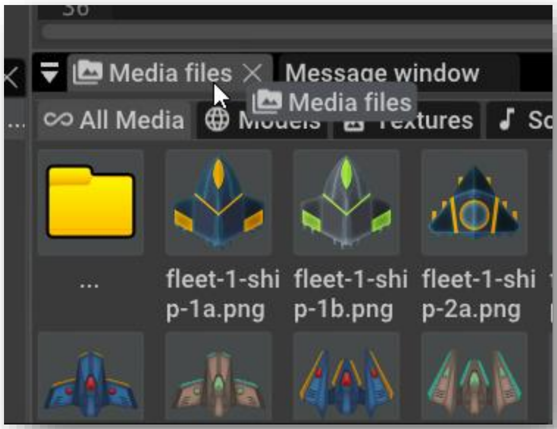
Integrated Help can be found on the right side.

At the base of the layout is the asset browser, media files viewer / message window and the media preview window.

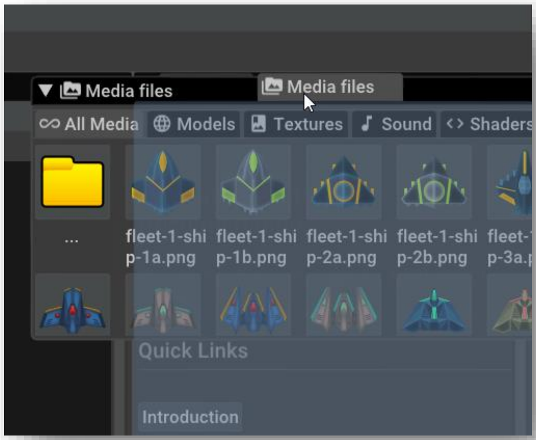
HOW TO CUSTOMISE THE UI

While we might feel this is the best layout, it's likely you'd disagree. The good news is that you can move around any of the windows and set up your own personalised design.

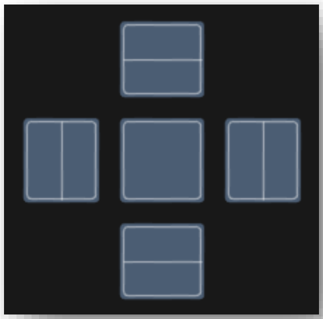
Let's imagine you want to move the Media Files section from the bottom and have it on the right column. All you need to do is to click and hold on the Media Files tab, as shown here:



Keeping the mouse click down, drag the Media Files tab over to the right and up to where the Help tab is. Once in the desired position you can let go of the mouse button and the tab will be in the new location.



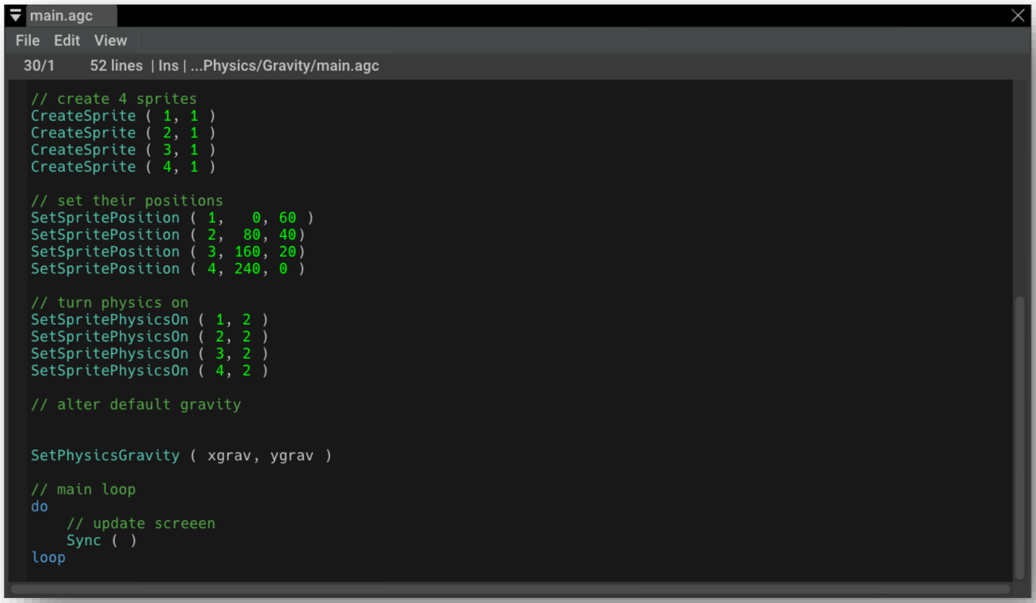
As you are dragging sections around you will see suggested slots where you can place a section.



Move the selection over these boxes and you will see a blue box indicate where the section would appear. Don't worry if you place it and it doesn't work for you, just drag it again and reposition. There's also a *Reset Layout* menu item in the View menu if you want to revert back to the default layout.

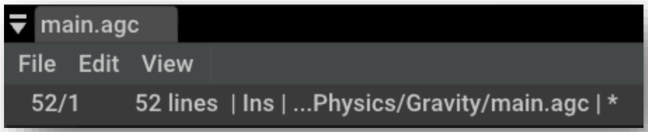
THE IDE (INTEGRATED DEVELOPMENT ENVIRONMENT)

This is where you will type in all your AppGameKit Script code. It's the heart of the product and the reason it's the front and centre of the experience. Here's the IDE showing some code script from the project *Gravity.apk* (one of the examples supplied). We'll break down the various areas so you know for sure what everything means.

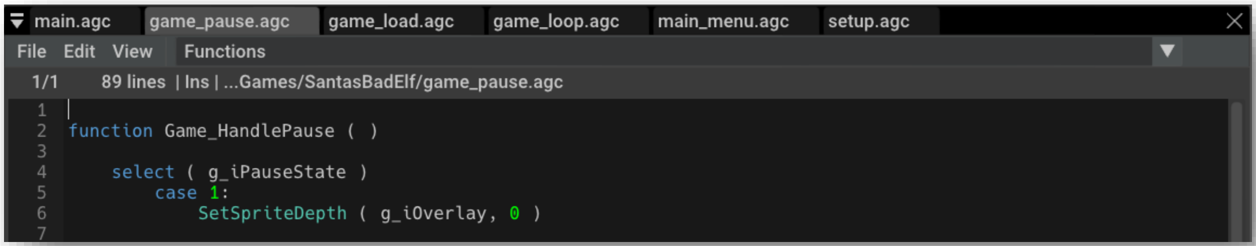


```
// create 4 sprites
CreateSprite ( 1, 1 )
CreateSprite ( 2, 1 )
CreateSprite ( 3, 1 )
CreateSprite ( 4, 1 )

// set their positions
SetSpritePosition ( 1,   0, 60 )
SetSpritePosition ( 2,  80, 40)
SetSpritePosition ( 3, 160, 20)
SetSpritePosition ( 4, 240, 0 )

// turn physics on
SetSpritePhysicsOn ( 1, 2 )
SetSpritePhysicsOn ( 2, 2 )
SetSpritePhysicsOn ( 3, 2 )
SetSpritePhysicsOn ( 4, 2 )

// alter default gravity


SetPhysicsGravity ( xgrav, ygrav )

// main loop
do
    // update screeen
    Sync ( )
loop
```

At the top of the IDE are tabs that allow you to move between different agc files.

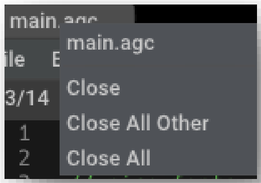


The above example only has one tab because it's just a project with one source file. Here's an example of a project with lots of source files. You can click between the tabs to view and edit each file:



```

function Game_HandlePause ( )

    select ( g_iPauseState )
        case 1:
            SetSpriteDepth ( g_iOverlay, 0 )

```

You can right click on the tab area to reveal a small pop up menu allowing you to close the current tab, all other tabs or all the tabs:



Below the tab line is a drop-down menu specifically for the IDE – File, Edit and View menus.

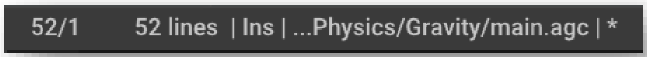
- The File menu just has Save, Save As and Close
- In the Edit menu you have:
    - Read-only mode – this is a toggle that once set stops you from making any edits. It useful for when you know you're done editing a source code file and don't want to mistakenly type text and mess up the file.
    - Find & Replace give you quick access to search
    - Toggle Line comment is a quick way to set a line into a comment line
    - Undo & Redo for correcting mistakes or changes
    - Cut, Copy, Paste, Select All for manipulating sections of code
    - Clear all breakpoints (see debugger section)
    - Clear all bookmarks – this removes all the set bookmarks in the current source file

Bookmarks

To set a bookmark in your code, just press Ctrl+"M"

To move between bookmarks, press Ctrl+"." And Ctrl+","

Under the drop-down menu there is some information texts:



52/1 52 lines | Ins | ...Physics/Gravity/main.agc | *

In the example above, the "52/1" refers to the line number the cursor is on and the character position from the left, in this case the cursor is on line 52 and on character position 1.

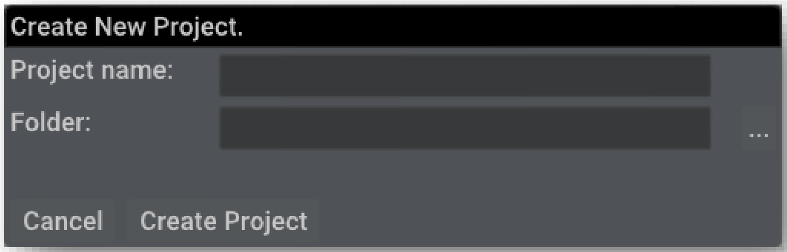
The "Ins" refers to the cursor mode – "Ins" means Insert and "Ovr" means Overwrite. You change the mode by pressing the "Ins" key on your keyboard.

The file path of the file is the next text in the line.

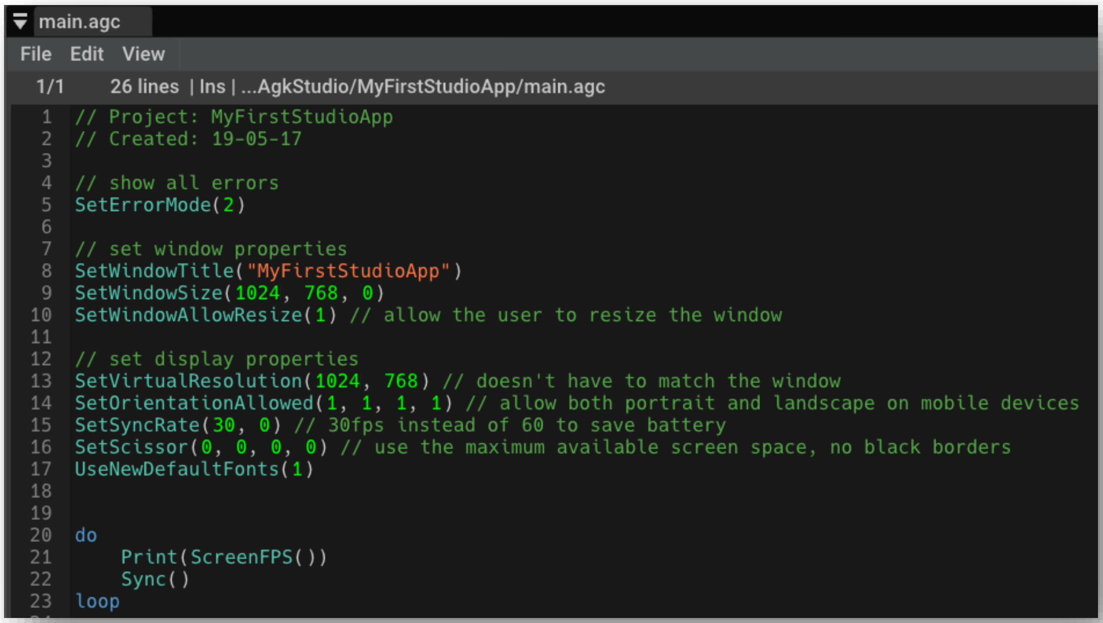
The "*" indicates the source code has had changes and the file is yet to be saved to store these changes. A quick File/Save will ensure the changes are saved and the "*" will disappear until the next change is made.

CREATING YOUR FIRST PROJECT

Let’s step through making a new project – the first thing to do is to choose File/New Project;



Type in the name of the project, for example *MyFirstStudioApp*, choose where you want the project to be stored (use the … to choose this with a file selector). Finally click on “Create Project”. The default files will be created and some initial code will appear in the IDE:



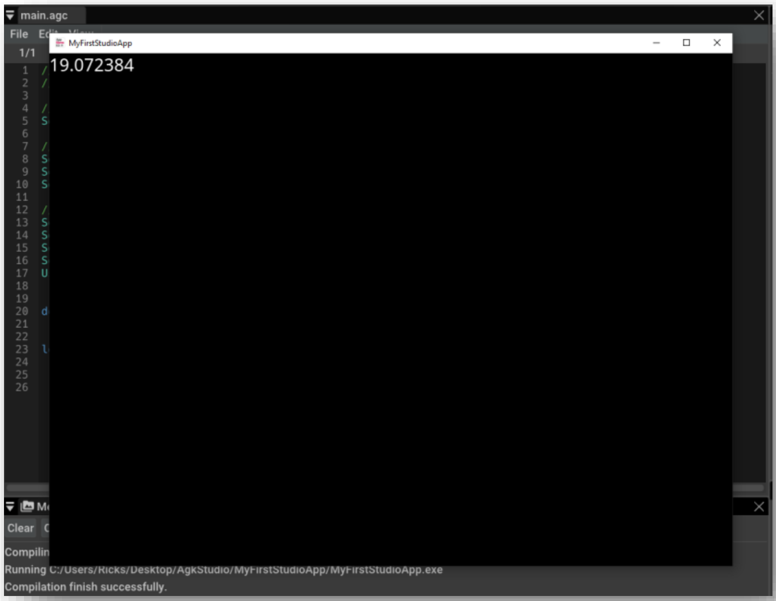
```
// Project: MyFirstStudioApp
// Created: 19-05-17

// show all errors
SetErrorMode(2)

// set window properties
SetWindowTitle("MyFirstStudioApp")
SetWindowSize(1024, 768, 0)
SetWindowAllowResize(1) // allow the user to resize the window

// set display properties
SetVirtualResolution(1024, 768) // doesn't have to match the window
SetOrientationAllowed(1, 1, 1, 1) // allow both portrait and landscape on mobile devices
SetSyncRate(30, 0) // 30fps instead of 60 to save battery
SetScissor(0, 0, 0, 0) // use the maximum available screen space, no black borders
UseNewDefaultFonts(1)


do
    Print(ScreenFPS())
    Sync()
loop
```

File Formats

Studio projects are stored in their own folders and include:

- apk project file
- agc source files
- media folder for assets

You can now Run this code using the Run icon in the toolbar. It will be compiled and run -this will result in a new window appearing like this:



```
19.072384
```

```
Compilin
Running C:/Users/Ricks/Desktop/AgkStudio/MyFirstStudioApp/MyFirstStudioApp.exe
Compilation finish successfully.
```

The last part of the code is looping around and printing the Screen FPS (Frames Per Second):

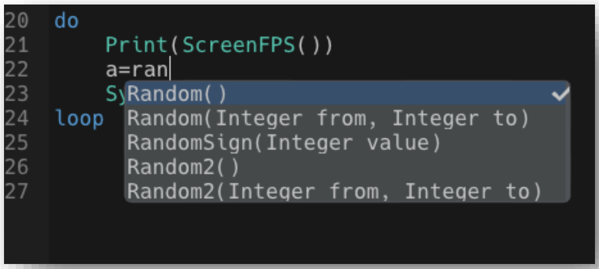
```
do
        Print(ScreenFPS())
        Sync()
loop
```

You can close the app by clicking the X in the top right or you can re-click the Play icon.

The IDE will help you by automatically formatting the code and comments as you type. If you wanted to add a comment above the "do" command in the code above you would insert a line and start typing "//" to indicate this is a comment line, once the second "/" is typed the line turns green because these starting characters indicate that this is to be a comment line.

The same applies to script commands. Let's add some code into the loop. As we type a=random(1,100), once the first three characters are typed "Ran", Studio will suggest possible commands from the thousands of commands that power the engine.

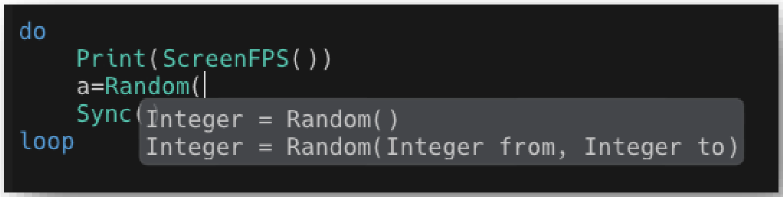
The more relevant command is at the top and has a tick icon to the right, if this is the command you want to use then you can press enter and the full Random text will appear where you are typing



```
do
    Print(ScreenFPS())
    a=ran
    SyRandom()
loop Random(Integer from, Integer to)
     RandomSign(Integer value)
     Random2()
     Random2(Integer from, Integer to)
```

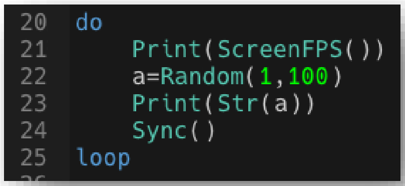
If you wanted a different command from that list you can use the up and down arrow keys on your keyboard to select them or mouse click a selection.

Commands usually have inputs to them and are listed between the brackets "()". In our example, once the first bracket is typed we are shown possible inputs – no inputs or between two integers:



```
do
    Print(ScreenFPS())
    a=Random(
    Sync(Integer = Random()
loop    Integer = Random(Integer from, Integer to)
```

Complete the code with two integer values and also add the line Print(str(a)) so it looks like this:



```
do
    Print(ScreenFPS())
    a=Random(1,100)
    Print(Str(a))
    Sync()
loop
```

Now Run the application and you will see two values being printed.

You might be wondering what the command "Str" means. If you place the cursor so it's positioned within the text of that command and then press the F1 key, you will get instant help about the command the cursor is over – Str().

Instant HELP!

Place the cursor over any command and then press F1. The Help window will show detailed help about that command. Hovering the mouse over a command

HOW TO LINK SOURCE FILES TOGETHER

Every project must have a *main.agc* source file. It's the first file that the compiler looks for when it starts the process of compiling the code of a project into bytecode format. You can have a very large main.agc if you want but doing this will become difficult to manage and navigate as it grows in size. A good practice is to split areas of a project into separate source files. Let's imagine a classic space invader screen, one way to split this game up might be as following;

- main.agc
- load_data.agc
- title_page.agc
- game_play.agc
- enemy_ai.agc
- player_control.agc
- sound_music.agc

To bring all these files together there are two special commands you can use in main.agc – *#insert* and *#include*.

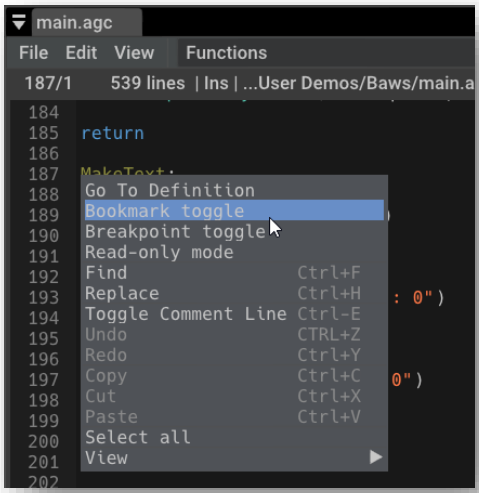
#insert will insert all the code from the chosen file directly at the point where the insert command is used.

#include just ensures the code is compiled along with main.agc and you can call any functions and routines from your code in main.agc. Here's some example of these commands:

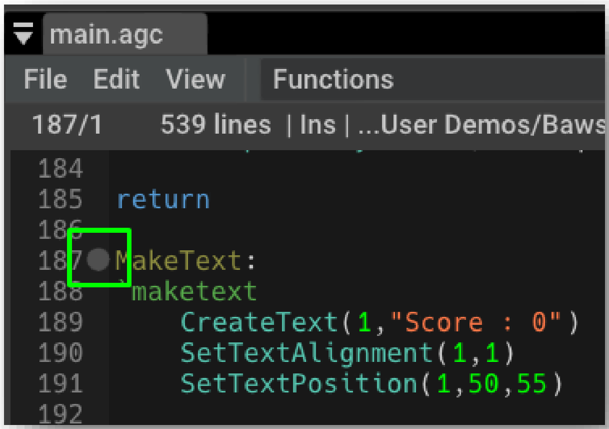
```
#include “load_data.agc”
#include “game_play.agc”
#include “enemy_ai.agc”
#insert “title_page.agc”
```

HOW TO SET A BOOKMARK

When source code files start to get large in line number terms it's useful to have a way to navigate to key areas. A bookmark provides such help and they are easy to set and access. To set a bookmark, place the cursor onto the line where you want the bookmark to be and right click the mouse button to reveal a menu of options. Select *Bookmark toggle* and a small grey dot will indicate that this line is now bookmarked:



```
return

MakeText:
```



```
return

MakeText:
 maketext
    CreateText(1,"Score : 0")
    SetTextAlignment(1,1)
    SetTextPosition(1,50,55)
```

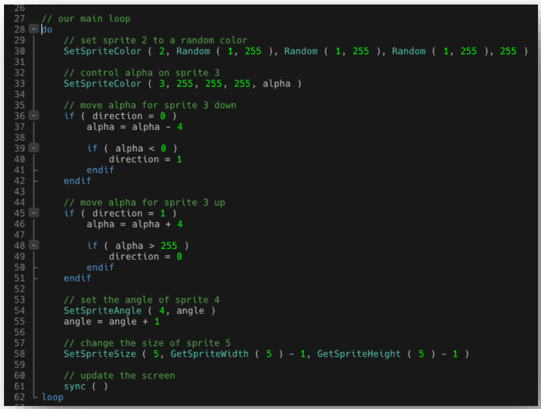
When you have more than one bookmark you can move the cursor between them by pressing Ctrl+"." and Ctrl+",".

CODE FOLDING

As projects get larger in size, they can be tricky to navigate around. To help with this you can make use of Code Folding. First you need to turn the feature on by selecting Edit/Preferences and then ticking Enable Code Folding in the Editor tab. If there's a project in the IDE you might see an immediate change as Code Folding switches on.

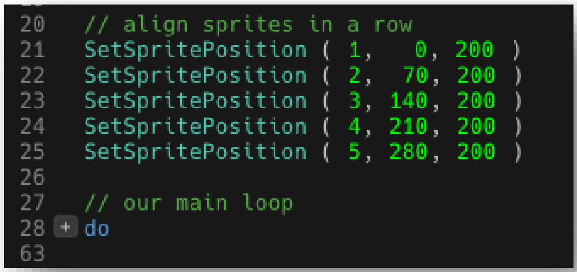
The IDE will not mark any loops or functions that it finds with a minus icon. Just click these minus icons and all code that comes within that area of code (marked by the start and end of that specific area) will be rolled up and hidden. What's left is a plus icon that you can click on to re-open the hidden code.

In this example a do…loop section of code can all be hidden from this:



```
// our main loop
do
    // set sprite 2 to a random color
    SetSpriteColor ( 2, Random ( 1, 255 ), Random ( 1, 255 ), Random ( 1, 255 ), 255 )

    // control alpha on sprite 3
    SetSpriteColor ( 3, 255, 255, 255, alpha )

    // move alpha for sprite 3 down
    if ( direction = 0 )
        alpha = alpha - 4

        if ( alpha < 0 )
            direction = 1
        endif
    endif

    // move alpha for sprite 3 up
    if ( direction = 1 )
        alpha = alpha + 4

        if ( alpha > 255 )
            direction = 0
        endif
    endif

    // set the angle of sprite 4
    SetSpriteAngle ( 4, angle )
    angle = angle + 1

    // change the size of sprite 5
    SetSpriteSize ( 5, GetSpriteWidth ( 5 ) - 1, GetSpriteHeight ( 5 ) - 1 )

    // update the screen
    sync ( )
loop
```

To just this:



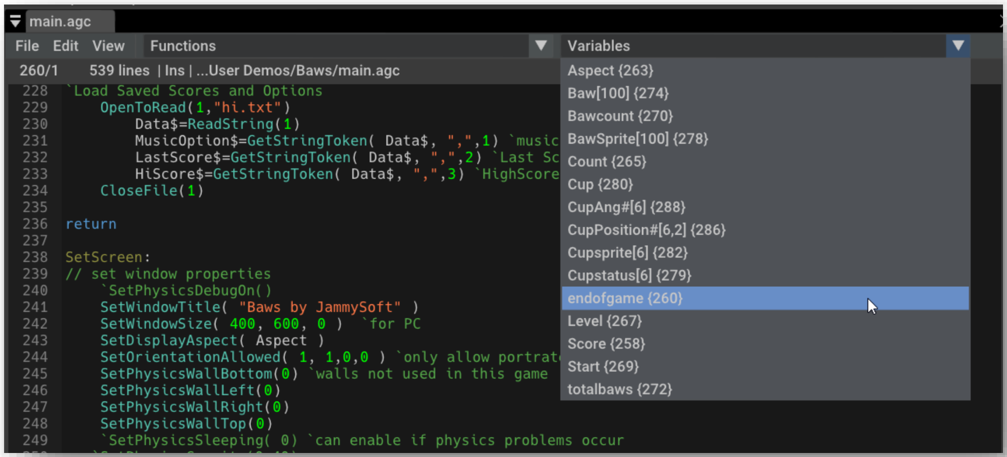
```
// align sprites in a row
SetSpritePosition ( 1,   0, 200 )
SetSpritePosition ( 2,  70, 200 )
SetSpritePosition ( 3, 140, 200 )
SetSpritePosition ( 4, 210, 200 )
SetSpritePosition ( 5, 280, 200 )

// our main loop
do
```

Some times you might want to hide a section of code that is not bound by a loop structure. In these cases, you can use the keywords `foldstart` and `foldend`.

Code folding will automatically work for these loop structure: `while - endwhile`, `do - loop`, `function - endfunction`, `remstart - remend`, `select - endselect`, `case - endcase`, `for - next`, `if - endif`, `repeat - until`.

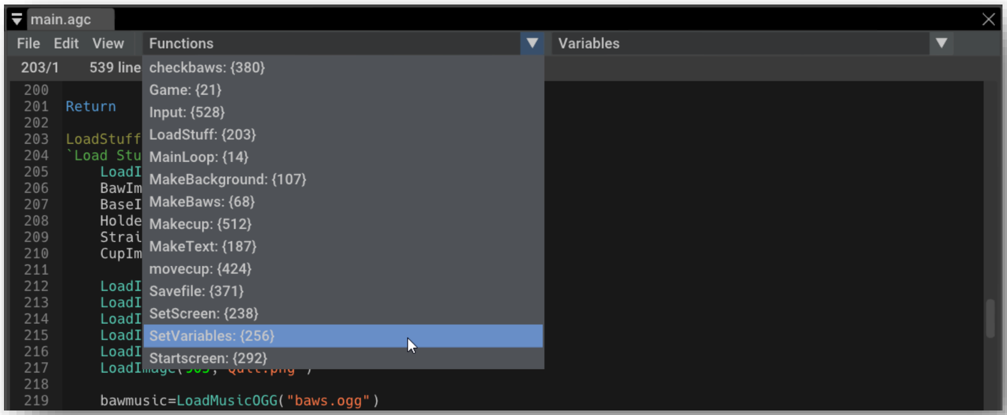
SYMBOLS LIST FOR FUNCTIONS & VARIABLES

As you become more proficient at programming, your projects are likely to become larger in size and could be hundreds and even thousands of lines long. The IDE has helpful tools to aid your navigation around your source code. When you start to add global variables and functions, the IDE will keep a record of these and makes an easy drop-down list for you to select from. In the image below you can see where these two lists are shown with the variables list open and the global variable *endofgame* selected. The number *{260}* indicates the line number where the variable is defined, so a click on that will take you to that point in the code.



```
`Load Saved Scores and Options
    OpenToRead(1,"hi.txt")
        Data$=ReadString(1)
        MusicOption$=GetStringToken( Data$, ",",1) `music
        LastScore$=GetStringToken( Data$, ",",2) `Last Sc
        HiScore$=GetStringToken( Data$, ",",3) `HighScore
    CloseFile(1)

return

SetScreen:
// set window properties
     `SetPhysicsDebugOn()
    SetWindowTitle( "Baws by JammySoft" )
    SetWindowSize( 400, 600, 0 )  `for PC
    SetDisplayAspect( Aspect )
    SetOrientationAllowed( 1, 1,0,0 ) `only allow portrat
    SetPhysicsWallBottom(0) `walls not used in this game
    SetPhysicsWallLeft(0)
    SetPhysicsWallRight(0)
    SetPhysicsWallTop(0)
     `SetPhysicsSleeping( 0) `can enable if physics problems occur
```

The same applies for functions, just click the menu and all the functions in this source file are listed. Click on the function and you're taken to the start of that function in the code.



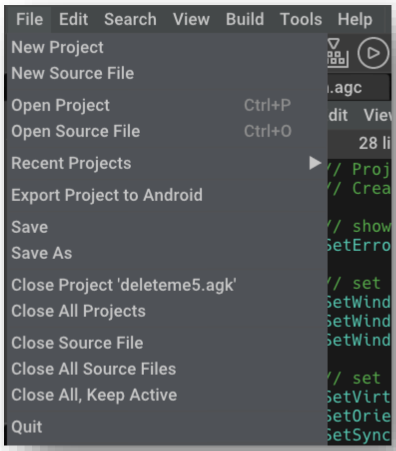
```

Return

LoadStuff
`Load Stu
    LoadI
    BawIm
    BaseI
    Holde
    Strai
    CupIm

    LoadI
    LoadI
    LoadI
    LoadI
    LoadI
    LoadI

    bawmusic=LoadMusicOGG("baws.ogg")
```

THE DROP-DOWN MENUS

At the top of the editor you will find the drop-down menus.

FILE MENU

Access all the features for creating, saving and exporting your projects with the File Menu.

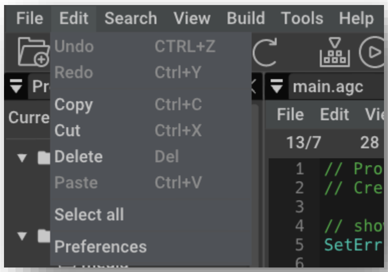


New Project	Use this to choose a name and where you want a new project to be stored. A file with the name you've chosen will be saved with the extension .apk.
New Source File	Projects have a main.agc which is the first source code file that is compiled by the AppGameKit Studio Compiler. You can *#include* and *#insert* other source files if you want to split your project into manageable sections. Use this menu item to create a new source file for the currently editable project.
Open Project	Use this to choose a previously saved project and start editing it.
Open Source File	Opens a source file into the main editing IDE.
Recent Projects	A time saving feature that shows a list of your most recently edited projects.
Export Projects to Android/iOS	If you want to publish a project then you will need to use this option to export the project so that it's compatible with Android and/or iOS app formats. You can only export iOS projects from the Mac version of AppGameKit Studio.
Save	Saves all the changes of the current project you're editing.
Save As	Let's you save the project with a different name. It's a good idea to save projects with different names from time to time to ensure you have a history of saved projects in case you delete a lot of code by mistake and decide later you wished you hadn't.
Close Project <current project name>	This closes down the currently edited project. Use Load Project to re-load it.

Close All Projects	All projects that are opened will be closed when you select this.
Close Source File	Closes the current source code file that is being edited.
Close All, Keep Active	This will close all the .agc files that may be open in the current project but it will leave open the active file that's shown in the IDE.
Quit	This closes the AppGameKit Studio application. If any files have been edited and are not saved, a warning will appear allowing you to save before you quit.

EDIT MENU

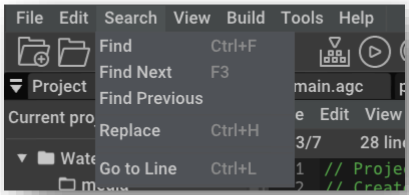
Manage and manipulate your code using the options in this menu.



Undo / Redo	These controls let you undo and redo any typing you have made so you can roll back where you were before.
Copy	Makes a copy of the selected text from the IDE into the cut buffer.
Cut	Cuts the selected text into the cut buffer and removes it from where it was cut from.
Delete	Deletes any selected text.
Paste	Pastes any text in the cut buffer at the place where the cursor is in the IDE.
Select All	This will highlight all the text in the current source file.
Preferences	Opens the preferences dialogue box, for details see the Preferences section.

SEARCH MENU

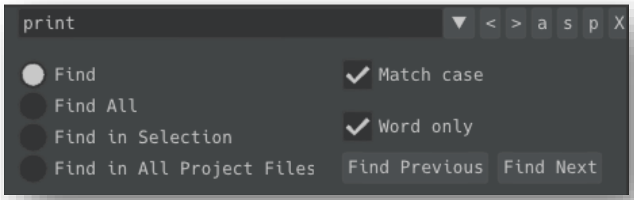
Use this menu to search through your source code.



Find	Opens a find edit box and will use any selected text at that time as the search text. Options allow you to find all references of the search string, within a selected area of code, across all the source files in the project.
Find Next	Finds the next occurrence of the search text.

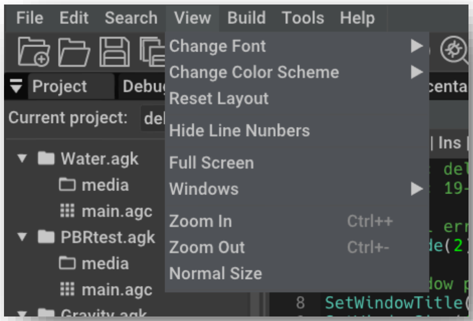
Find Previous	Finds the previous search string.
Replace	Allows you to replace the search text with a replacement text. You can replace just one at a time or all references found and within an area of code or across all source files. Flags can be set to only find the text if all the letter cases match and if the search term is only found as a word on its own
Go to Line	Moves the IDE cursor to the line of your choosing.

If you press Control+Shift+F you can call up the advanced search box:



VIEW MENU

Personalise the IDE to your liking with these features in the View menu.

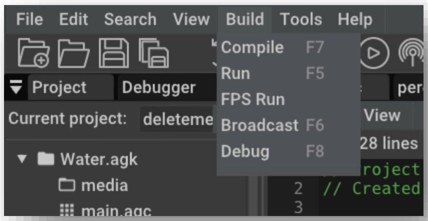


Change Font	In this sub menu you can change the font size and the font style used by the main IDE program listing text. You can also change the font size of the IDE texts (menus, dialogue box texts etc). With screen sizes having many different resolutions this is a feature you should use to make the best choice for your setup.
Change Color Scheme	Here you can set the color scheme that best suits how you like to view the editor. There is a choice of colour schemes for the main editors and one for the IDE where you spend a lot of time doing the coding. You can also create random colour schemes by using the Style Generator in preferences.
Reset Layout	If you have moved windows, hidden some and want to revert back to the default layout then just use this to reset the windows back to their starting positions.
Hide Line Numbers	A simple toggle to turn the code line numbers off and on.
Full Screen	To maximise the available screen area, you can use this feature. The surrounding window edges are hidden leaving you with more room to see more of your code, scenes etc. Press F11 to come back out of full screen mode.
Windows	You might not want all the various windows showing when you are coding. Here you can hide/show them as you see fit.

Zoom In	Zooms the IDE code larger each time you select this. You can also do this on quick keys with Control and +
Zoom Out	Zooms the IDE code smaller, quick key is Control and -
Normal Size	Normal size resets the IDE code to the default setting.

BUILD MENU

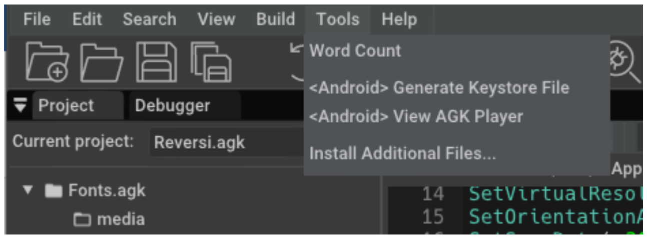
This menu is for Running, testing and broadcasting your projects.



Compile	After loading a project or coding your own, use this option to instruct AppGameKit Studio to take your source code files and compile them into byte code. If there are no errors in your code then the message window will report “Compilation finished successfully”. If there were errors detected then the message window will list them and they will be highlighted in the code at each line with a red marker. Fix the issues and then try to re-compile.
Run	The compiler in AppGameKit Studio is super-fast and will be almost immediate. In most cases you will want to compile and then run your project to see your latest changes. If you use “Run” your project is compiled and if there are no errors your project is run and you will see the project start.
FPS Run	This special run mode will put the IDE to sleep while your app is being run, meaning you get the best possible frame per second. It’s as if you were running the compiled version directly.
Broadcast	If you want your projects to work on devices like iPhones, iPads and Android phones then you will want to use this feature. It allows you to broadcast the project over Wi-Fi to the devices you want to test on. First make sure your AppGameKit Studio is on the same Wi-Fi as your devices. The devices you are testing on need to be running the AppGameKit Player app. You can download these from the Apple App Store, the Google Play Store and the Amazon App Store. You will find direct links to these apps in the Help menu. Once you have everything setup you then press Broadcast and your app will be compiled and then sent out via Wi-Fi and picked up by the player apps. You might need to allow AppGameKit Studio to gain access via your firewall too. See the Broadcast section for more details.
Debug	When you code, you will at times make mistakes. Some are very easy to fix but others can be very hard to find and fix. In these cases, you need the help of a debugger tool. AppGameKit Studio has a powerful debugger, providing ways to watch the changes in variable values, step line by line through your code, run and show variables in an auto update mode. Read the Debugger section for full details.

TOOLS MENU

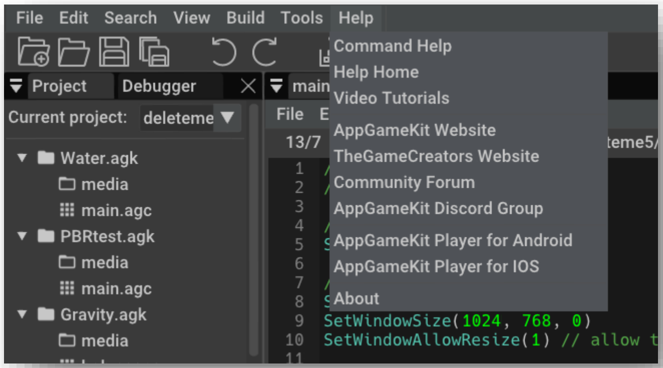
A small tools menu providing a word count and keystore creation.



Word Count	Shows you the total number of words in the currently viewed source code file.
<Android> Generate Keystore File	If you plan to submit your project to the Google or Amazon apps stores then you will need to create a Keystore file. This is a private key that signs your apps when you submit them or update them on these app stores. It's **very important** to note that the same Keystore file has to be used every time you update your apps. More details are given in the dialogue box when you access this feature. Once you press the Generate button your keystore file will be created in the location chosen in the Output File Location field.
<Android> View AGK Player	As part of the AppGameKit Studio installation we have provided the player app in Android APK format. Select this and the file location will be shown in an explorer window. You can then copy the file and side load it onto any Android device you want to use with AppGameKit Studio's broadcast feature. You can also just download the latest player app from the Google App Store and the Amazon App Store.
Install Additional Files…	AppGameKit Studio comes with a set of example projects for the Tier1 Script language. There's also a set of native libraries for development with C++. As a default these are located in the AppGameKit Studio installation folder. You might prefer to store them elsewhere - in the case of Windows the Program Files folder is not writeable so it's not suitable for projects. Using this feature, you can copy these additional files to a location of your choice.

HELP MENU

Help and tutorial resources are all at hand from this menu.

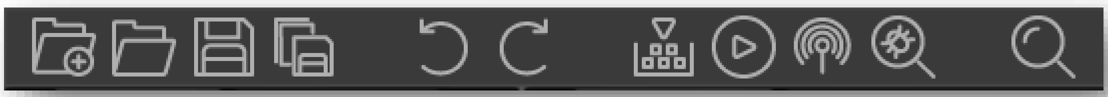


```
SetWindowSize(1024, 768, 0)
SetWindowAllowResize(1) // allow t
```

Command Help	This changes the Help window to show the main command help section. Note that there's a search facility at the top of this section, start typing a command name and you will see lots of suggestions appear as you type.
Help Home	Shows the home page in the Help window.
Video Tutorials	Opens your browser to show the AppGameKit video tutorials. Many of these were created using AppGameKit Classic but they are still relevant and will work fine for Studio.
AppGameKit Website	Opens the AppGameKit main website.
TheGameCreators Website	Opens TheGameCreators website.
Community Forum	Takes you to *TheGameCreators* forums. There are thousands of registered users in the community and here we encourage friendly help and sharing of ideas. If you're stuck with a coding issue then we highly recommend you sign up and post into the forums, there's always a knowledgeable user willing to help.
AppGameKit Discord Group	If you like to chat all things AppGameKit Studio then this is the place to be. Live chat and various channels are waiting to welcome you!
AppGameKit Player for Android	Links you to the Android Player app on the Google App Store.
AppGameKit Player for iOS	Takes you to the Apple App Store where you can download the iOS player app.
About	Shows information about the team who created AppGameKit Studio and also shows the End User License Agreement.

TOOL BAR ICONS

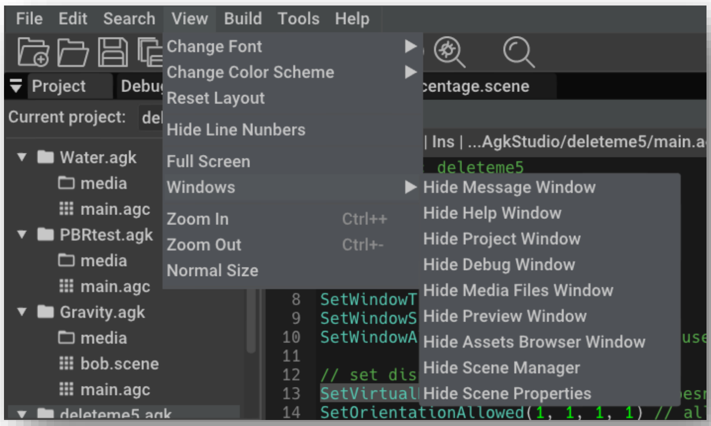
At the top of the screen is the tool bar where you have quick access to regular features that you will use many times when coding in AppGameKit Studio.



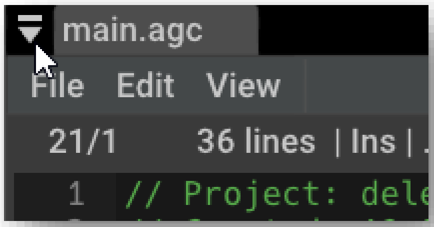
	New Project – Creates a new project based on the name you enter.
	Save Project - Saves the current project with all recent changes.
	Save the current file - Saves just the changes made to the currently edited agc file.
	Save all open files – Saves all the .agc files that are currently open in the IDE.
	Undo/ Redo - Lets you undo and redo multiple edits in your typing.
	Compiler - Compiles the current project with the results shown in the Message window.
	Run - Compiles and runs the current project.
	Broadcast - Compiles and then Broadcasts the current project over the Wi-Fi. You will need a device running the player app for this to work.
	Debug - Compiles and then runs a debug session.
	Find - Brings up the search box.

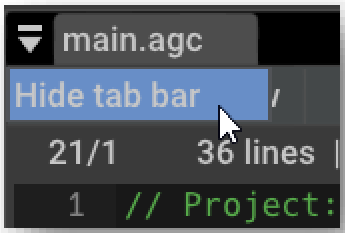
THE UI WINDOWS

AppGameKit Studio has a number of windows that help you organise your projects. You have complete control if you want them shown or to make them hidden. To hide a window, use the View/Windows menu:

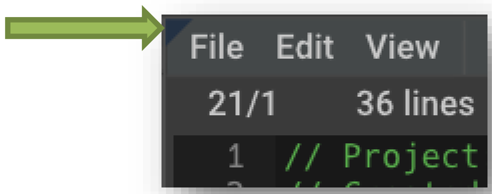


It's also possible to show and hide a window if it's the only item in the tabbed list. To hide a window, click on the down arrow in the top left and then a one-line menu will appear *Hide tab bar*:



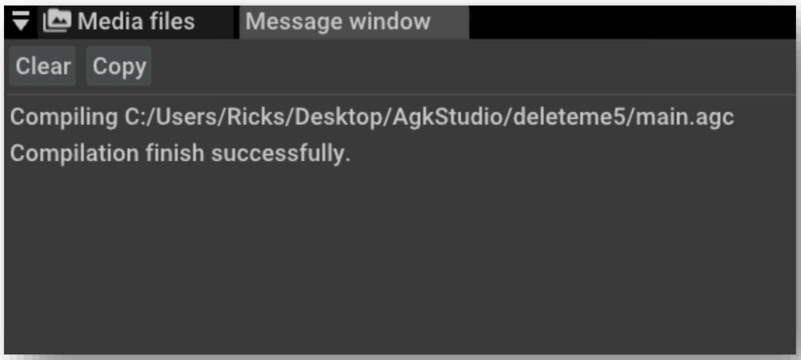


To re-show the hidden window you just click the very top left where there is a very subtle edge:



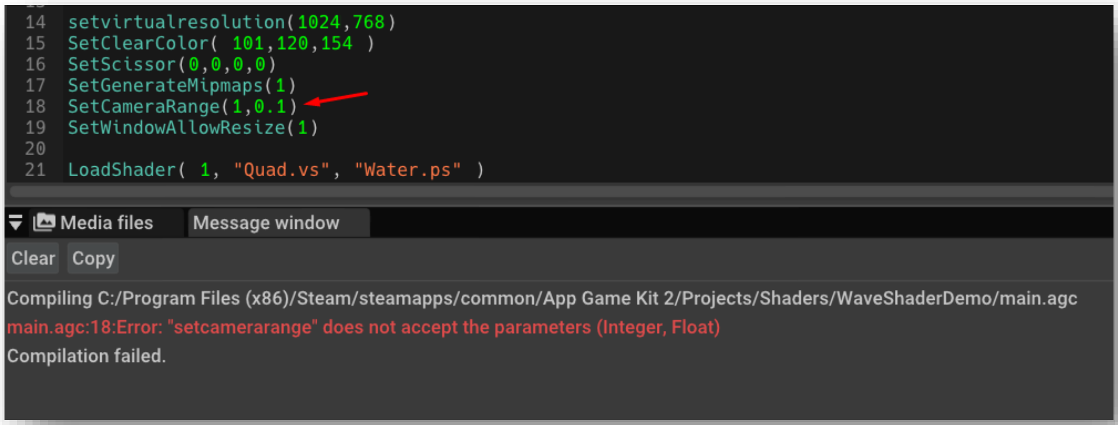
MESSAGE WINDOW

The purpose of this window is to give result feedback from project compiles. In this example the project "deleteme5" has compiled successfully.



```
Compiling C:/Users/Ricks/Desktop/AgkStudio/deleteme5/main.agc
Compilation finish successfully.
```

In this next example the message window shows *Compilation Failed* due to an error line that is referenced in the red text report "main.agc 18:Error "setcamerarange" does not accept the parameters (Integer, Float)". The line in the code needs an extra parameter adding to the command and then it will compile.



```
14  setvirtualresolution(1024,768)
15  SetClearColor( 101,120,154 )
16  SetScissor(0,0,0,0)
17  SetGenerateMipmaps(1)
18  SetCameraRange(1,0.1)
19  SetWindowAllowResize(1)
20
21  LoadShader( 1, "Quad.vs", "Water.ps" )
```

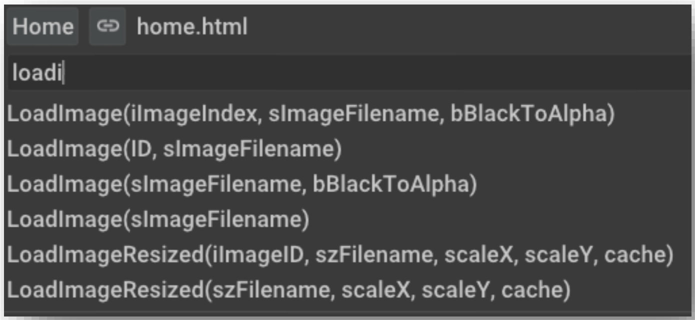
```
Compiling C:/Program Files (x86)/Steam/steamapps/common/App Game Kit 2/Projects/Shaders/WaveShaderDemo/main.agc
main.agc:18:Error: "setcamerarange" does not accept the parameters (Integer, Float)
Compilation failed.
```

HELP WINDOW

Help is always at hand and is fully integrated into AppGameKit Studio.

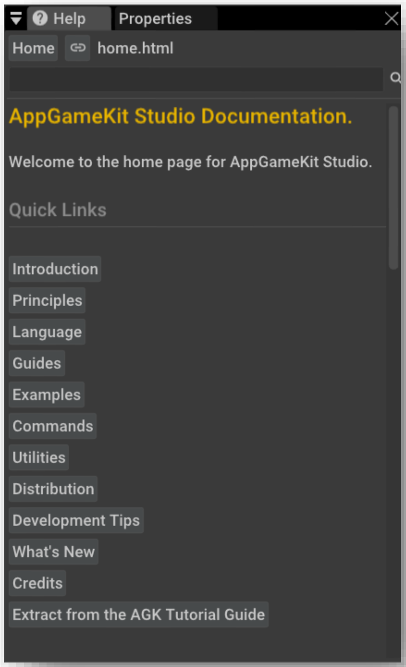
There are hundreds of commands in AppGameKit Studio and all of them are documented in the help. It's not easy remembering command names so we've introduced a command search in the Help window. Just start typing the beginning of the command name and a list of suggestions will appear from which you can pick. As you type more characters, the list of suggestions will update showing a more relevant list from which to choose.

Below is an example where we're typing "loadi", all commands that start with this series of characters is shown below. Many times, you will half remember a command name and this tool will help you find the full name.



LoadImage(iImageIndex, sImageFilename, bBlackToAlpha)
LoadImage(ID, sImageFilename)
LoadImage(sImageFilename, bBlackToAlpha)
LoadImage(sImageFilename)
LoadImageResized(iImageID, szFilename, scaleX, scaleY, cache)
LoadImageResized(szFilename, scaleX, scaleY, cache)

The Help provides details about the main language structure of AppGameKit Studio, guides, examples, app distribution and many other areas of advice. It's well worth checking out all the menus in the help.



AppGameKit Studio Documentation.

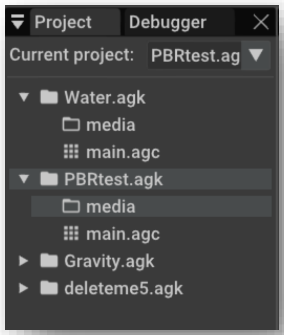
Welcome to the home page for AppGameKit Studio.

YouTube Tutorials!

You can learn to code with our online tutorials on YouTube. We're always adding new videos and advice so please Subscribe here.

PROJECT WINDOW

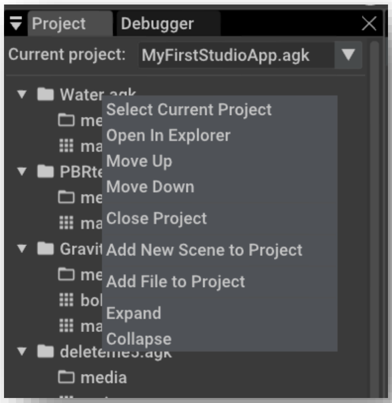
The project window provides an organised list of all your projects. At the top of the window it shows which project is the current one you are editing. You can change to a different project by clicking on the down arrow and choosing a different one or you can right click on a project title in the list and choose "Select Current Project".



From the File Menu, use the New or Open Project menu options to add new projects to the list.

On Windows you can drag and drop AppGameKit Project and source files into this area.

When you right click on a project name in this list a range of choices appear.



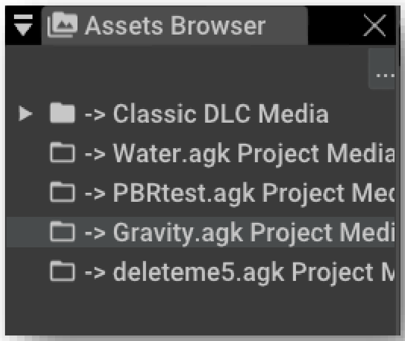
Select Current Project	Select this to make the project the current project that's being edited.
Open In Explorer	Open the file location of the project in a file explorer.
Move Up/Move Down	Move the project up or down in the list of projects.
Close Project	Close the project, removing it from the list.
Add New Scene to Project	Add a new scene to this project. For details about scenes see the Scene Editor chapter.
Add File to Project	Add a .agc file to the project.
Expand/Collapse	Open up and display all the files of the project / Close up the list and just show the project name.

ASSET BROWSER WINDOW

The Asset Browser window works in collaboration with the Media Files Window. This window shows a list of file path locations that point to where your project asset files are located for all the open projects.

You can add extra folders to the list by clicking on the "…" icon top right and you will be presented with a file explorer. Choose the directory and it will be added to the list.

The folder structure can be explored by clicking on the right pointing arrows, providing access to assets that may be many folders down from the starting path.



-> Classic DLC Media
-> Water.agk Project Media
-> PBRtest.agk Project Med
-> Gravity.agk Project Medi
-> deleteme5.agk Project M

When you close a project, the list will remove the folder for that project from the list.

Clicking on a path location will then change what the Media Files Window shows.

AppGameKit Classic DLCs

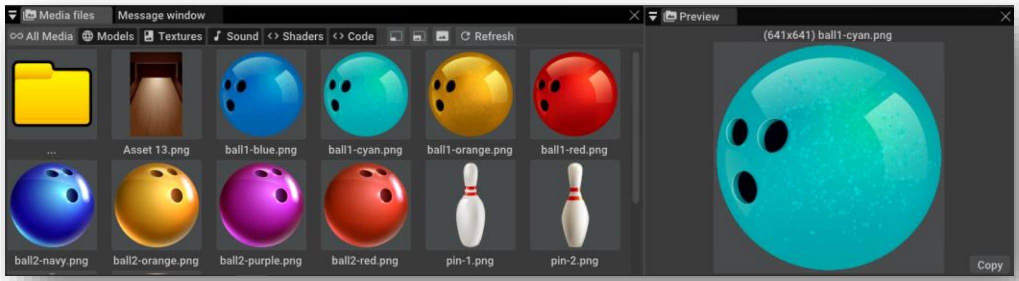
If you own asset libraries from the original AppGameKit Classic (Giant Asset Packs, Sound Library etc) you can tick a box in the *Preferences/IDE* and the folder for those assets will be added to the list.

MEDIA FILES WINDOW & PREVIEW WINDOW

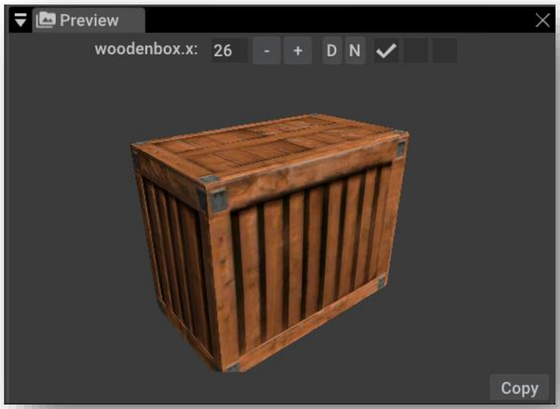
This window shows a visual representation of the assets in the selected folder chosen in the Asset Browser window. The window defaults to a tab that shows all media types. Sometimes you might want to focus on one type of asset so there are tabs for textures, sounds, shaders and code.

The window has three thumbnail icons to let you choose the size that best suits your display. If the folder contents change you can update the list by pressing the Refresh button.

If you click on an asset in the window it will be shown in the preview window.



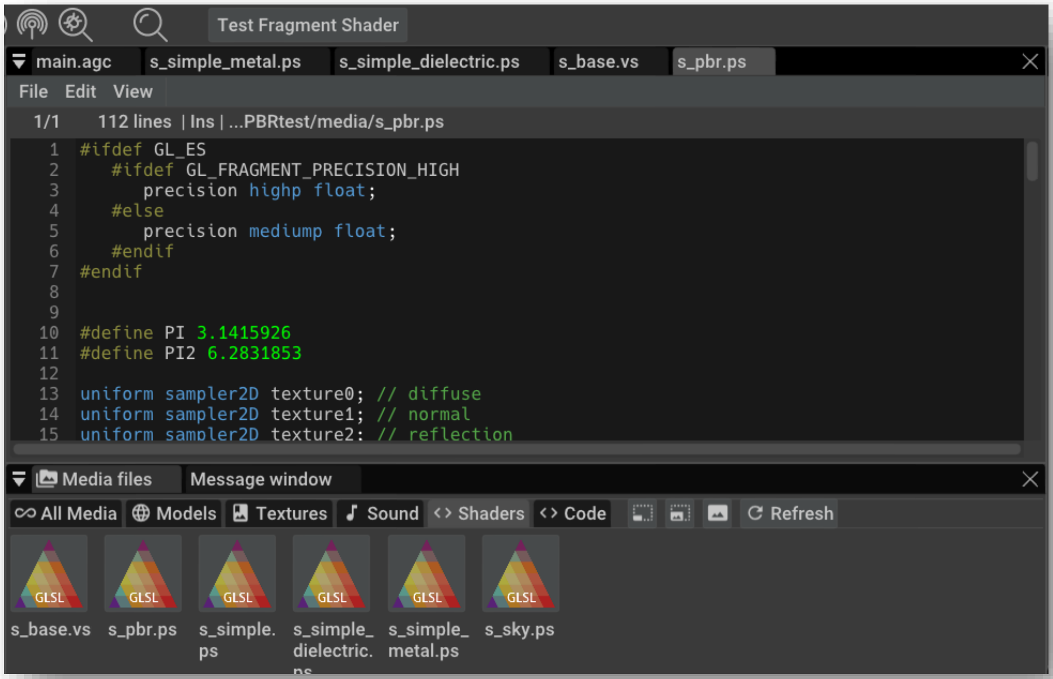
Sound files can be double clicked on so you can listen to them before deciding to add them to your project. The files will show with a special play icon so they are easy to recognise.



3D objects can also be previewed and you can copy any relevant diffuse and normal maps across to the D and N icons to visualise them in the preview. You can zoom in and out using the + and – buttons or the mouse wheel and there are toggles to rotate the object in X, Y and Z axis.

Text files can also be viewed and there’s a mode for viewing files in hexadecimal format.

Shader files are special GPU programs that can perform graphical effects when you run your app. You can preview and edit the code of a shader in the IDE. They are displayed with a GLSL triangle shape in the browser. Double click on them and you can edit them in the IDE and in the tool bar you'll see a "Test Fragment Shader" button – click this to check a shader compiles fine. If it doesn't an error will be shown and (if you have a deep knowledge of shader coding) you can debug the shader.



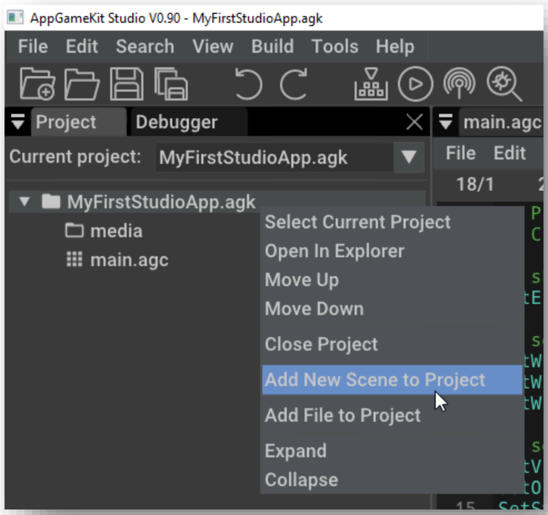
```
#ifdef GL_ES
   #ifdef GL_FRAGMENT_PRECISION_HIGH
      precision highp float;
   #else
      precision mediump float;
   #endif
#endif


#define PI 3.1415926
#define PI2 6.2831853

uniform sampler2D texture0; // diffuse
uniform sampler2D texture1; // normal
uniform sampler2D texture2; // reflection
```

THE SCENE EDITOR

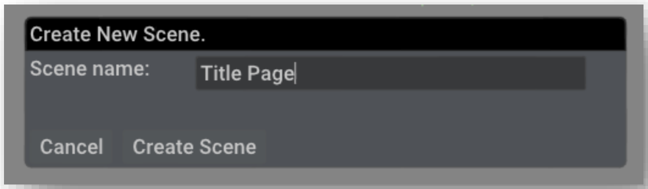
Built into the heart of AppGameKit Studio is the Scene Editor. This specialised tool is designed to help with your UI design work when creating apps. Instead of having to guess the co-ordinate positions of screen assets like sprites and text, you can position them into place and see exactly where they will appear.

HOW TO CREATE A SCENE IN A PROJECT

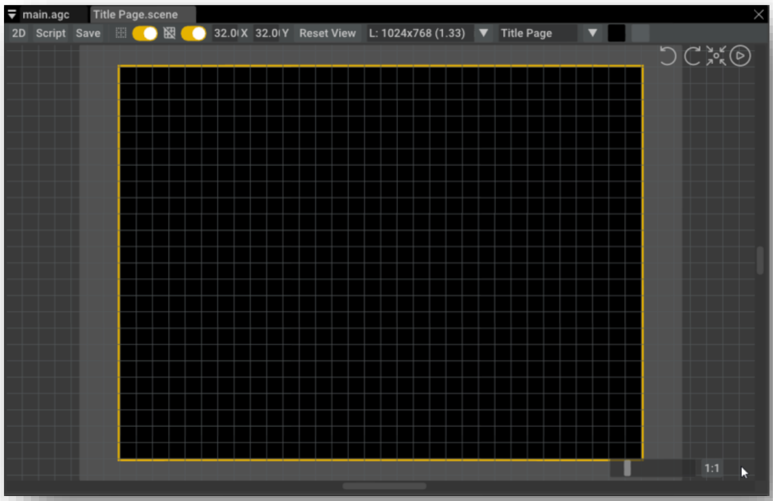
The Scene Editor will appear when you decide you want to add a scene to your project. To do this you simply right click on the project name in the Project window. A menu will appear in which "Add New Scene to Project" is available to choose, like so;



You will then be prompted to give the scene a name;



The main window where the IDE is displayed will now show a new scene tab and it will look like this;

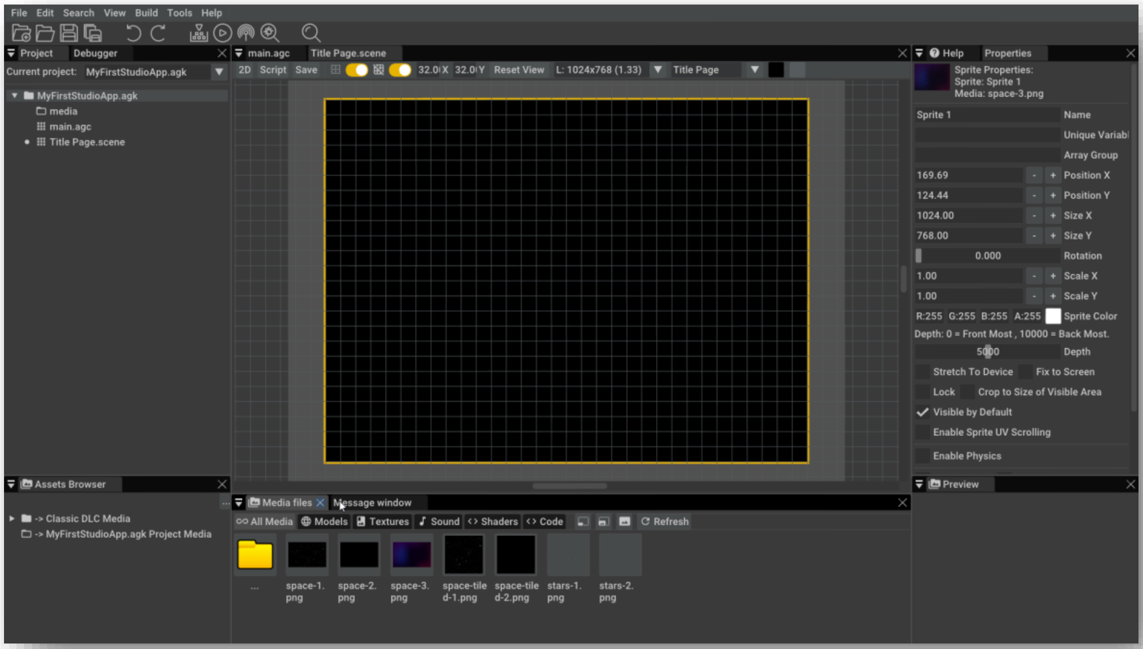


Across the top are some tool bar gadgets, here's what they do.

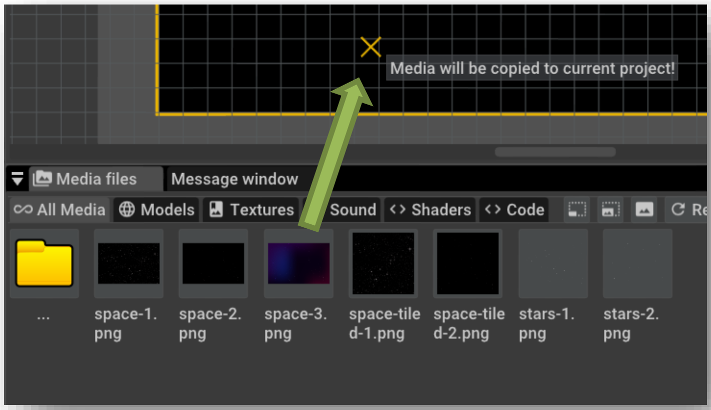
Script / 2D toggle	Click this and you will see all the automated script code that the Scene Editor creates as you use it. Unless you really know what you're doing it's best not to edit this script. Click again and the Scene Editor will be displayed (as above).
Save	Saves any changes you have made.
▦ ●	This toggles the yellow outline on and off. Anything inside the area will be seen when used on a device. Items outside are off screen.
▨ ●	Grid on/off toggle.
32.0(X 32.0(Y	The size of the grid (default is 32x32).
Reset View	Re-centres the view of the scene.
L: 1024x768 (1.33) ▼	The resolution of the scene. Click the down arrow to choose a different resolution. The L stands for Landscape and P for Portrait.
Title Page ▼	The scene being edited. You can choose a different scene with the drop-down arrow and you can add more scenes to this scene.
■ ■	Background and grid colours. Click on the coloured boxes to choose a different colour.

HOW TO ADD MEDIA INTO A SCENE

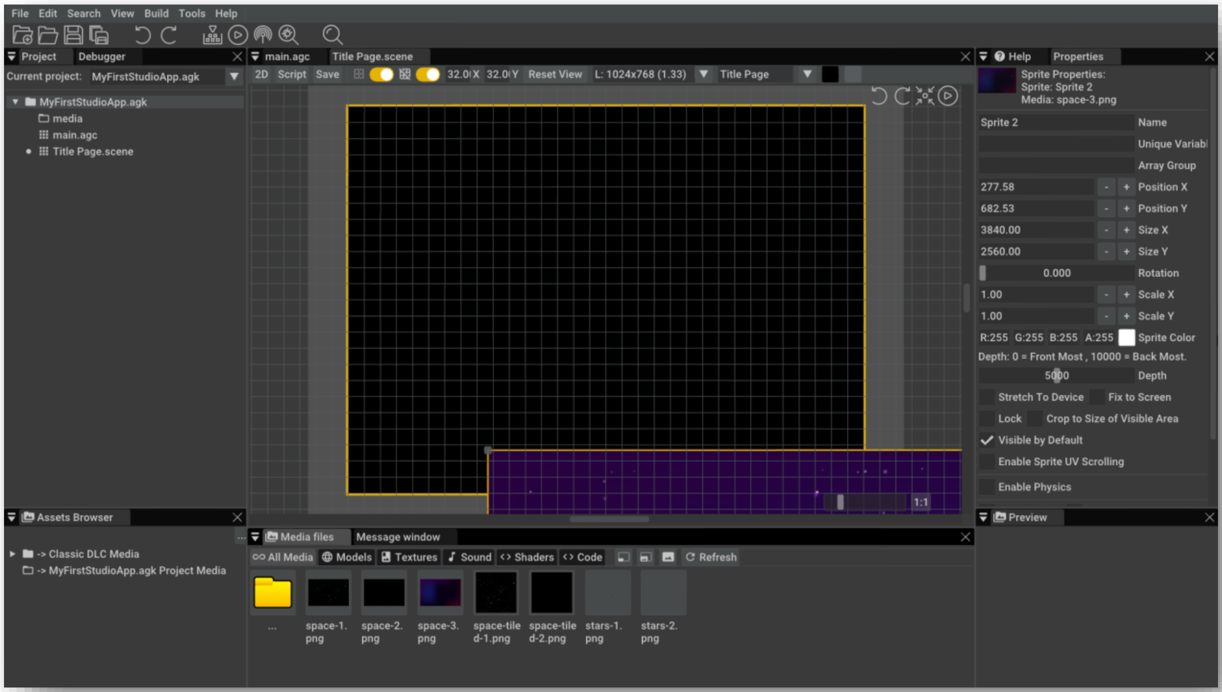
Using the Asset Browser and Media Files windows you can drag & drop media into your scene. In the example below we're viewing media from the AppGameKit Classic Giant Asset Pack DLC.



To drag & drop, left click and hold down on to an image and move the mouse dragging the image into the scene. It will display as a yellow X until you let go of the left mouse button to drop it. For example:


Media will be copied to current project!

Depending where you drop the image and the size of the image you will see something like this;



Sprite Properties:
Sprite: Sprite 2
Media: space-3.png

Sprite 2	Name
	Unique Variabl
	Array Group
277.58	Position X
682.53	Position Y
3840.00	Size X
2560.00	Size Y
0.000	Rotation
1.00	Scale X
1.00	Scale Y
R:255 G:255 B:255 A:255	Sprite Color

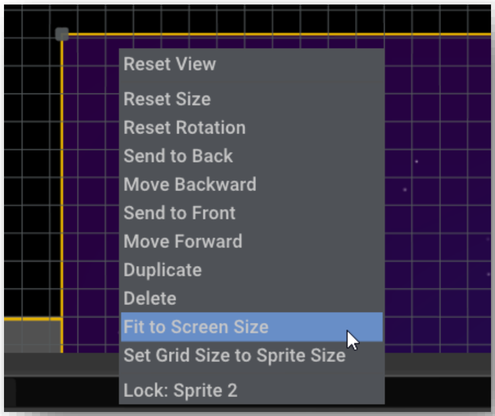
Depth: 0 = Front Most , 10000 = Back Most.

5000	Depth

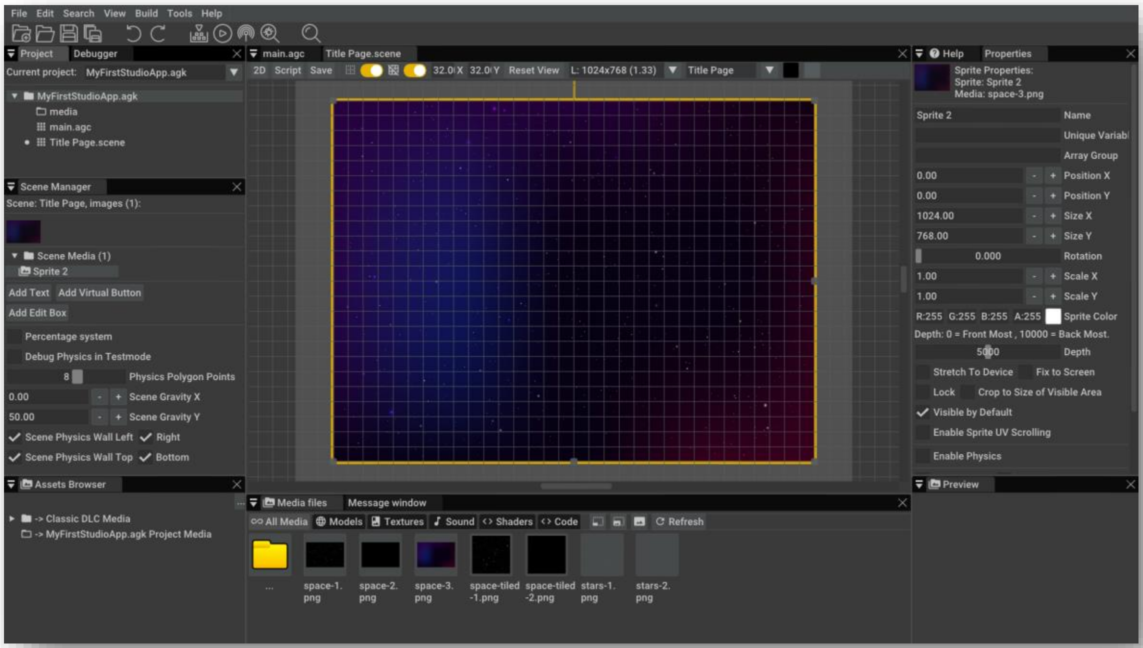
Stretch To Device
Fix to Screen
Lock
Crop to Size of Visible Area
✓ Visible by Default
Enable Sprite UV Scrolling
Enable Physics

32.0 X 32.0 Y L: 1024x768 (1.33)

As this image we have dragged in is a backdrop, we can right click on the image and choose to Fit to Screen Size.



Now the image takes up the full background;



Current project: MyFirstStudioApp.agk

- MyFirstStudioApp.agk
  - media
  - main.agc
  - Title Page.scene

32.0 X 32.0 Y L: 1024x768 (1.33) Title Page

Scene: Title Page, images (1):

- Scene Media (1)
  - Sprite 2

Percentage system
Debug Physics in Testmode
8 Physics Polygon Points
0.00 Scene Gravity X
50.00 Scene Gravity Y
✓ Scene Physics Wall Left ✓ Right
✓ Scene Physics Wall Top ✓ Bottom

Sprite Properties:
Sprite: Sprite 2
Media: space-3.png

Sprite 2	Name
	Unique Variabl
	Array Group
0.00	Position X
0.00	Position Y
1024.00	Size X
768.00	Size Y
0.000	Rotation
1.00	Scale X
1.00	Scale Y
R:255 G:255 B:255 A:255	Sprite Color
5000	Depth

Depth: 0 = Front Most , 10000 = Back Most.

Stretch To Device Fix to Screen
Lock Crop to Size of Visible Area
✓ Visible by Default
Enable Sprite UV Scrolling
Enable Physics

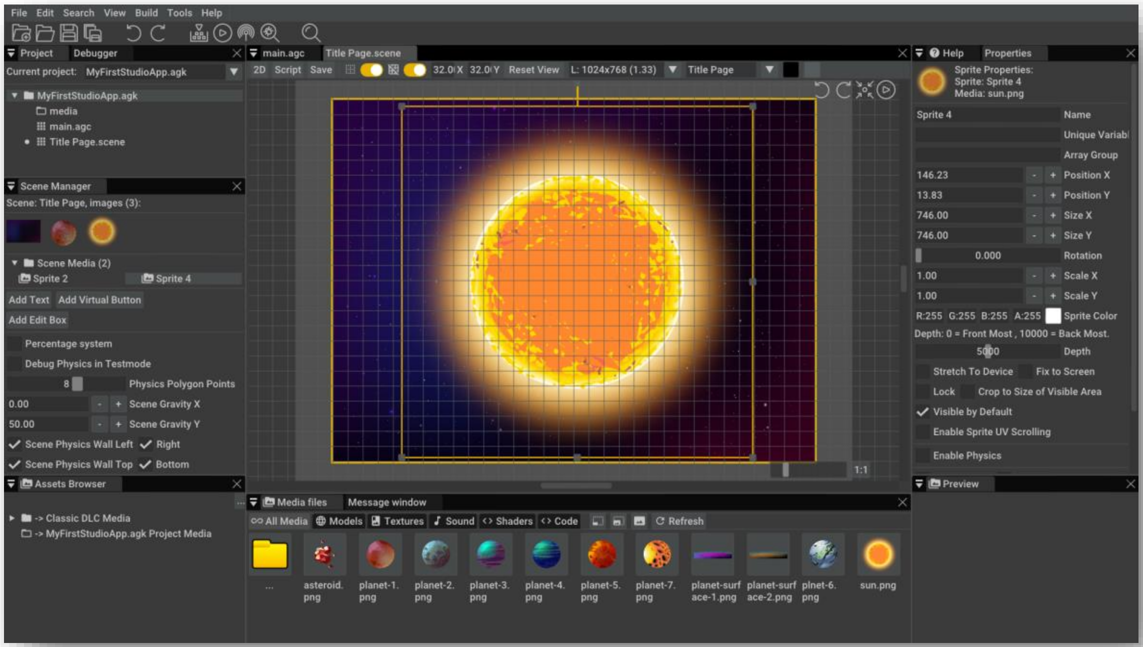
... space-1.png space-2.png space-3.png space-tiled-1.png space-tiled-2.png stars-1.png stars-2.png

It's important to note that if the media has come from a location that's separate from your project's media folder then AppGameKit Studio will copy across the media into the project media folder.

Notice that there are now some extra windows showing. On the left is the Scene Manager tab and on the right a Properties tab is showing. The scene manager lists all the objects added to the scene, lets you create text, virtual buttons and edit boxes and it sets the 2D physics for the scene. The object properties tab manages all the options for the currently selected object.

SCALING, ROTATING AND MOVING OBJECTS

We'll now add some more objects into the scene. Here we've added a sun graphic:



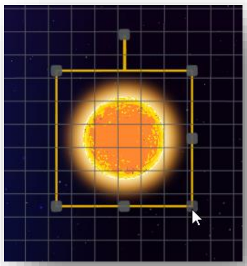
Sprite 4	Name
	Unique Variabl
	Array Group
146.23	Position X
13.83	Position Y
746.00	Size X
746.00	Size Y
0.000	Rotation
1.00	Scale X
1.00	Scale Y
R:255 G:255 B:255 A:255	Sprite Color
5000	Depth

Depth: 0 = Front Most , 10000 = Back Most.

Stretch To Device, Fix to Screen, Lock, Crop to Size of Visible Area, ✓ Visible by Default, Enable Sprite UV Scrolling, Enable Physics

Physics Polygon Points: 8; Scene Gravity X: 0.00; Scene Gravity Y: 50.00; ✓ Scene Physics Wall Left; ✓ Right; ✓ Scene Physics Wall Top; ✓ Bottom

This is too big! We want to scale the size of the object down. The yellow box around the object indicates that it's the currently selected object. Using the grey boxes on each corner we can scale the sun to the size we desire.

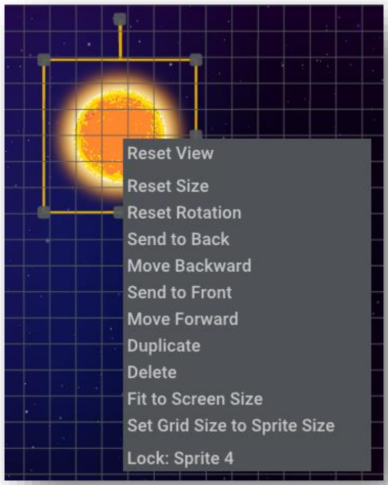


The aspect ratio of the graphic will be retained as we scale the sun image. If we wanted to make it thinner then we can use the grey button that is half way down the right yellow line. It's also possible to rotate the image with the grey button that's at very top.

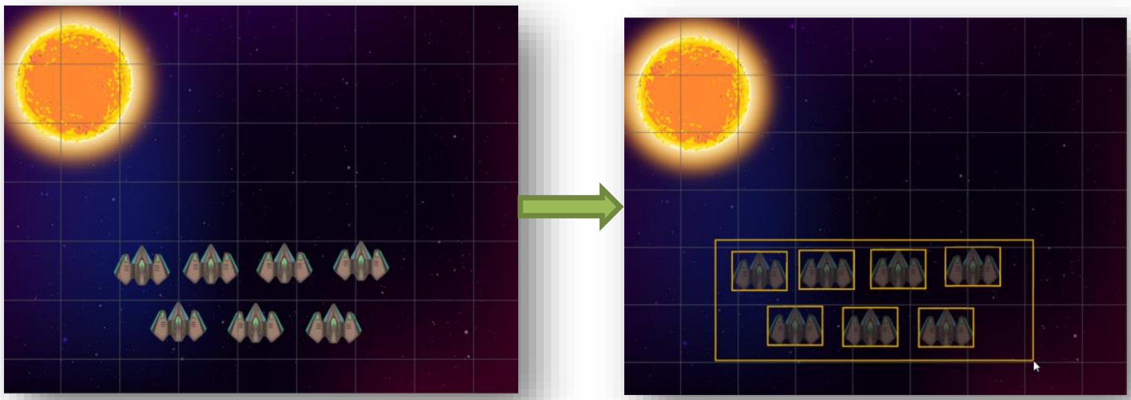
To move the sun object, you just need to click and drag it. You can also use the arrow keys to move the currently highlighted object for fine single pixel movement.

As we mentioned above, you can right click on an object and a drop-down menu will appear. Let's look at all the options of this menu.

Reset View	This re-centre's the Scene Editor if you've moved around or zoomed the view.
Reset Size	The object will be re-sized to its original size.
Reset Rotation	The rotation of the object is reset to zero degrees.
Send to Back	Sets the object to the first image to be drawn in the scene (i.e. behind everything else).
Move Backward	Moves the object one place back in the draw order list.
Send to Front	Makes this object the last one drawn (in front of all others).
Move Forward	Brings the object one place up the draw order.
Duplicate	Makes a duplicate of the object with all the same resizing and orientation settings.
Delete	Remove the object from the scene.
Fit to Screen Size	Resizes the object so it matches the size of the scene.
Set Grid Size to Sprite Size	Sets the grid to use the same X and Y sizes of the object. If you have a tile set you want to make a level with then this is a quick way to set the grid size.
Lock	Locks the object so it cannot be moved by accident. To unlock, right click where the object is and a menu will appear giving you an option to unlock it.



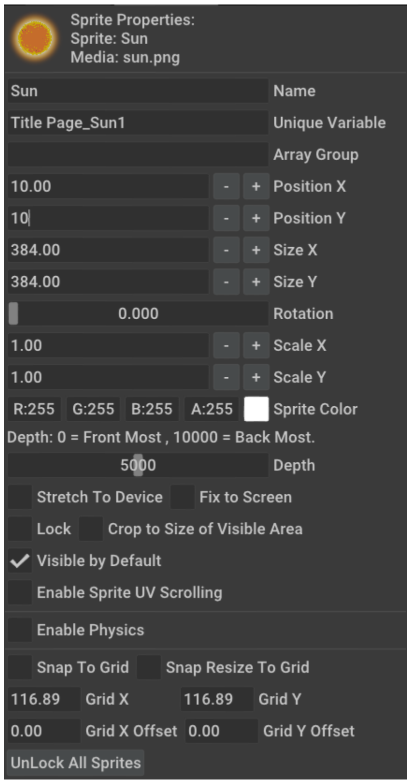
In this updated scene below there are some space ship objects. The blue space backdrop has been locked so as to avoid selecting it when moving other objects. If we now want to move the ships to a new position it would be tedious to move each one individually. Using a rubber-band technique you can select all the objects you want to group together and then move them in unison.



SPRITE PROPERTIES

Each time you highlight an object in the Scene Editor the properties window will update to show all the relevant data for that sprite. Let's detail all the properties you can edit.

Name	It's useful to give a sprite a name that makes sense and something you can relate to later. In this case we have called it "Sun".
Unique Variable	If you want to reference the sprite in your code then you must assign a name for a Global variable. Here we typed in Sun1 and then Studio updated it with the name of the scene "Title Page_Sun1".
Array Group	If you want to reference a group of sprites (say aliens) then you can assign an array with this field.
Position X,Y	The X and Y position of the sprite.
Size X,Y	The X,Y size of the sprite in pixels.
Rotation	The rotation of the sprite (0-360 degrees).
Scale X,Y	The X,Y scale of the sprite.
RGB Sprite Colour	The Red, Green, Blue and Alpha colour settings. Great for giving a sprite a colour tint.
Depth	This is the draw depth of the sprite. The depth ordering works from zero (front) to 10,000 (back).
Stretch to Device	Use this to scale the sprite to full X and Y width of the scene (ideal for backdrop images).
Fix to Screen	If your game scrolls then you might not want items like HUD images to move (score displays, maps etc). Tick this and the sprite will always stay in the same place.
Lock	Locks the sprite so you cannot easily move it by mistake.
Crop to Size of Visible Area	Viewing a large image that displays into the offscreen area can get in the way. Tick this to hide the area that is off screen.
Visible by Default	If you don't want this sprite to show at the start of the app then un-tick this.
Enable Sprite UV Scrolling	Allows you to modify the UV parameters of a sprite. For a demo see this video.
Enable Physics	Let's you turn on physics for the object (see below).
Snap to Grid	Snaps the movement of the sprite to the X,Y grid.
Snap Resize to Grid	With this on, resizing will snap to the grid sizes.
Grid X, Y	The X and Y size of the grid.
Grid X, Y Offsets	Sometimes you want a grid to line up with a graphic on screen, use these to fine tune where the grid starts.
Unlock All Sprites	A quick way to unlock all locked sprites.



Sprite Properties:
Sprite: Sun
Media: sun.png

Sun	Name
Title Page_Sun1	Unique Variable
	Array Group
10.00	Position X
10	Position Y
384.00	Size X
384.00	Size Y
0.000	Rotation
1.00	Scale X
1.00	Scale Y
R:255 G:255 B:255 A:255	Sprite Color

Depth: 0 = Front Most , 10000 = Back Most.

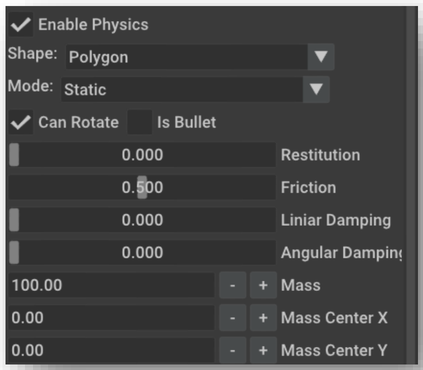
5000	Depth

Stretch To Device · Fix to Screen
Lock · Crop to Size of Visible Area
✓ Visible by Default
Enable Sprite UV Scrolling
Enable Physics
Snap To Grid · Snap Resize To Grid
116.89 Grid X · 116.89 Grid Y
0.00 Grid X Offset · 0.00 Grid Y Offset
UnLock All Sprites

SPRITE PHYSICS PROPERTIES

If you want certain sprites to have physics properties you can tick the "Enable Physics" flag and physics will be applied to the sprite at run time. You will also be shown extra physics properties you can set;

Shape	This defines the type of collision shape the sprite will use in the physics simulation. The options are No Shape, Circle, Box and Polygon.
Mode	This setting sets the type of physics mode the sprite will work under. Options are Static, Dynamic and Kinematic.
Can Rotate	If set to on, this allows the object to rotate as it collides with other physics objects. There may be times when you don't want that to happen and so you can stop it by setting this to No.
Is Bullet	A "bullet" in physics is set on an object that moves really fast and needs the collision detection all the time. So, ticking this property will put this object into this mode.
Restitution	This is a value between 0 and 1. The higher the value, the more bounce the object will have.
Friction	Defines how grippy or slippery the sprite will be. A low value makes it grip more, whereas a high value will make it more like it's on ice.
Linear Damping	This creates a resistance and slows down the effects of the physics. The higher the number, the more resistance is applied.
Angular Damping	A high value here will slow down the effects of any rotation.
Mass	The mass weight of the object. A balloon would have a very low mass. A larger object such as a rock could be set to 100 for example.
Mass Center X, Y	Defines where in the sprites shape the center of the mass is positioned. Setting this properly based on the image will have a significant effect on results.



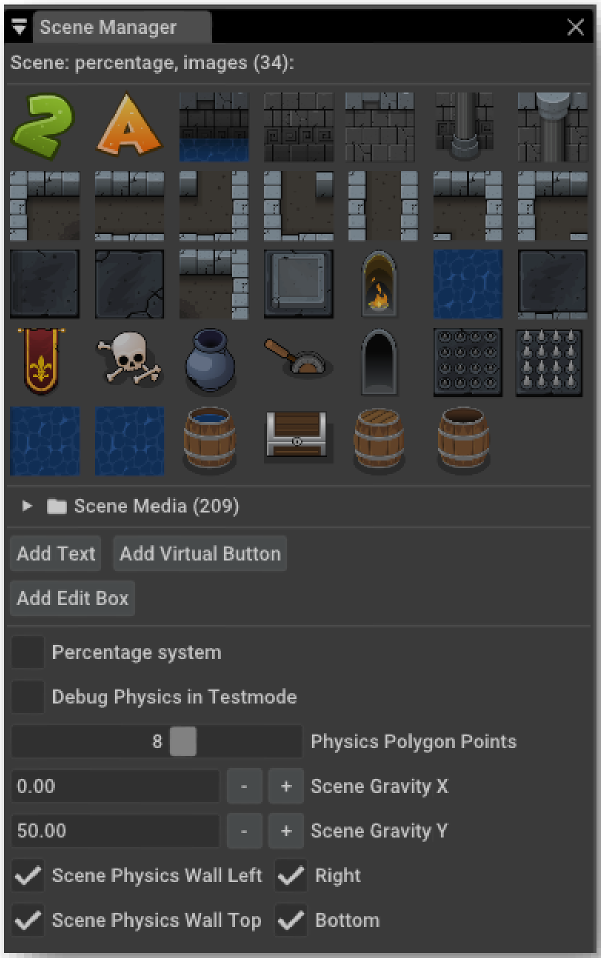
✓ Enable Physics
Shape: Polygon
Mode: Static
✓ Can Rotate Is Bullet
0.000 Restitution
0.500 Friction
0.000 Liniar Damping
0.000 Angular Dampin
100.00 - + Mass
0.00 - + Mass Center X
0.00 - + Mass Center Y

SCENE MANAGER

The scene manager keeps a record of all the game assets you have added to the currently selected scene. As you drop sprites into the scene a thumbnail will appear in this window. If you click on one of these thumbnails you enter a draw mode state (see below).

There are buttons for creating text, virtual buttons and edit boxes. Simply click on them and the object will appear in the top left of the scene ready for you to edit it.

Other properties of the scene can also be set in this window and these are detailed in the table below.

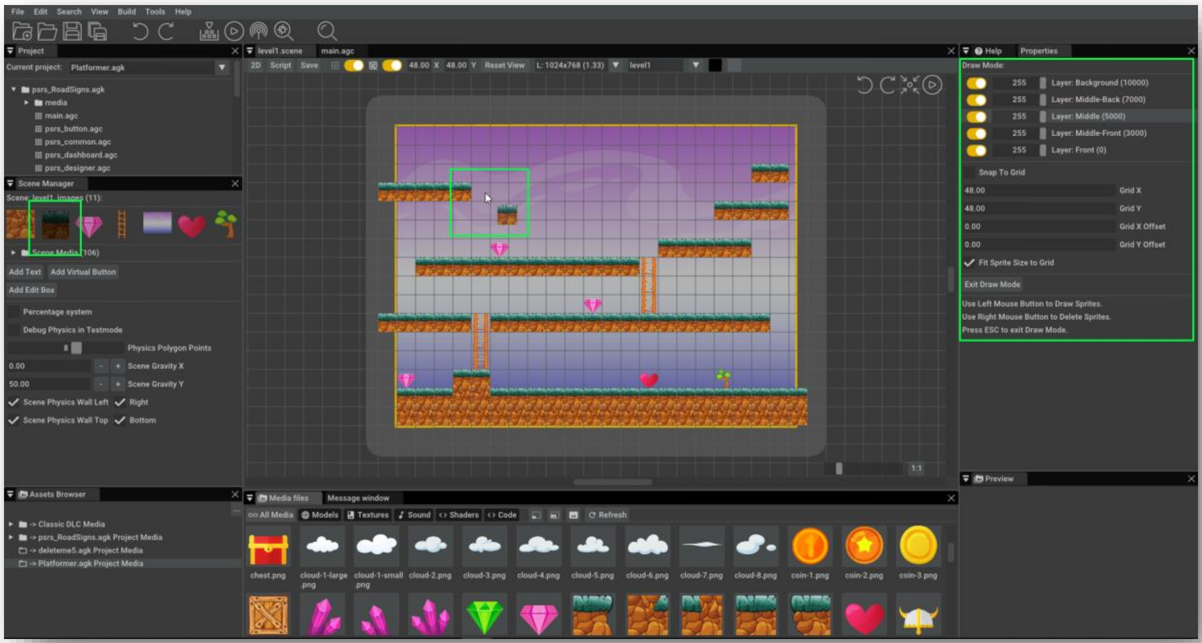


Scene	Shows the name of the scene and how many images are being used in it.
Thumbnails	Thumbnails of the images in the scene are shown here. If you click on an image you enter into draw mode, allowing you to quickly draw with an image into the scene.
Scene Media	Shows a text list of all the items in the scene. If you click on one of these then the item will be highlighted in the scene and you can edit its size, position, rotation and other properties.
Percentage system	Tick this box if you want the scene to work with the percentage system (co-ordinates use 0-100). If left unticked it will use virtual resolution (actual screen resolution).
Debug Physics in Testmode	The Scene Editor has a quick test mode. When you play a scene that has physics sprites in it and this is ticked, the physics collision shape will show around the objects to help you debug.
Physics Polygon Points	Defines how many polygons should be used in the polygon collision shape around sprites. Less polygons are more efficient but could result in poor collisions.
Scene Gravity X / Y	This sets the gravity of the scene in both the X and Y axis. Try playing around with these numbers to see some interesting results!
Scene Physics Wall Left/Right/Top/Bottom	An invisible wall will be placed around the scene when any of these are ticked. It's used to contain sprites within the visible scene but there are times when you might want objects to pass out of the bounds.

DRAW MODE

When designing a scene there are many times where you'll need a quick way to place down tile-based images to build up a larger image. This is the purpose of this special mode.

To enter draw mode, you simply select the image you want to draw with from the scene manager. That image will be shown in a small thumbnail to the bottom right of the mouse pointer. You can then click and draw the image into the scene and keep repeating this until you want to choose another image. To the right-hand side, the properties window shows details and options that you can use in Draw Mode.



Draw Mode:

●	255	Layer: Background (10000)
●	255	Layer: Middle-Back (7000)
●	255	Layer: Middle (5000)
●	255	Layer: Middle-Front (3000)
●	255	Layer: Front (0)

Snap To Grid

48.00	Grid X
48.00	Grid Y
0.00	Grid X Offset
0.00	Grid Y Offset

✓ Fit Sprite Size to Grid

Exit Draw Mode

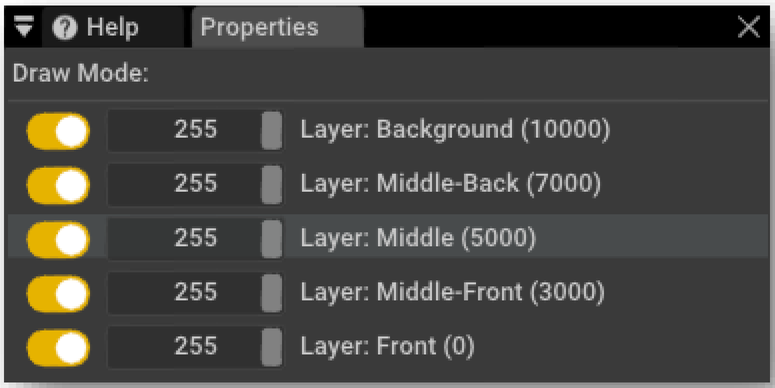
Use Left Mouse Button to Draw Sprites.
Use Right Mouse Button to Delete Sprites.
Press ESC to exit Draw Mode.

If you want to ensure your images fit and work to the grid you have setup then you must ensure you turn on *Snap to Grid* and if the image is smaller or larger than the grid cells, they can be scaled with *Fit Sprite to Grid.*

You can exit grid mode by clicking on the image in the scene manager or clicking on *Exit Draw Mode* in the Properties window.

USING LAYERS IN DRAW MODE

When you enter Draw Mode a simple layer system will appear in the Properties window that you can take advantage of when drawing sprites, text and edit boxes into the scene.



●	255	Layer: Background (10000)
●	255	Layer: Middle-Back (7000)
●	255	Layer: Middle (5000)
●	255	Layer: Middle-Front (3000)
●	255	Layer: Front (0)

There are five yellow buttons representing different layer sections of the draw depth of the scene. If these buttons are on, then any sprites drawn will be shown in the scene. The value to the right of these buttons is an alpha valve (normally set to 255 – fully visible). By reducing the alpha value, you can fade a layer to see more clearly the other layers.

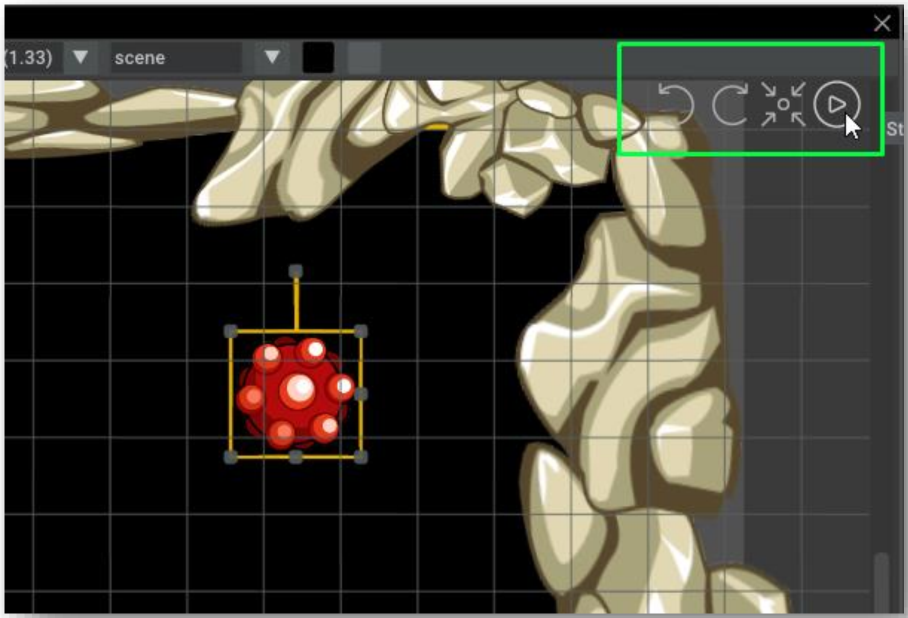
The layer system requires you to manage your scenes carefully. If you place a backdrop down you should set its depth property to 10,000. You then decide which images are to be placed in the middle-back, middle, middle-front and front areas of the scene.

TOP TIP!

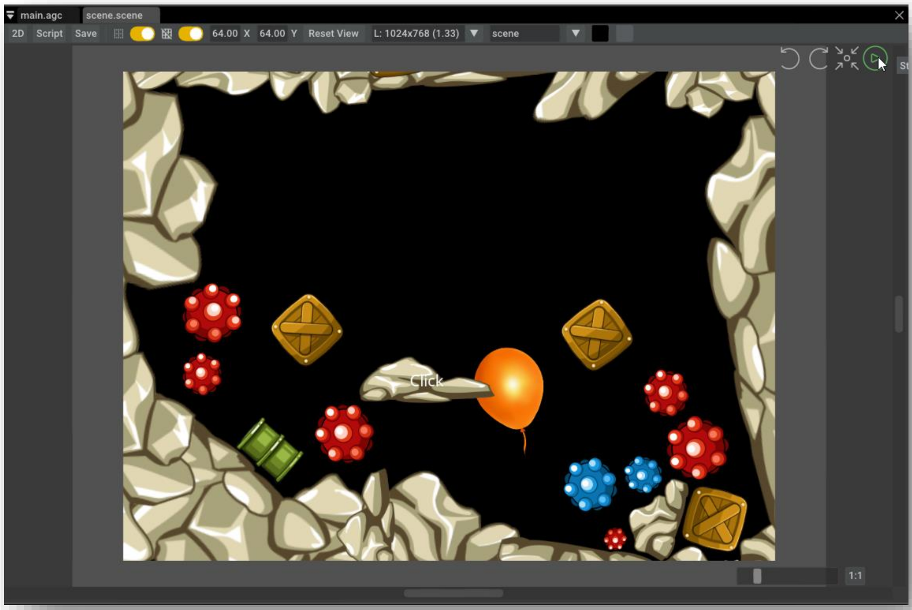
Hold Shift and click & drag to move around a scene with a rubber band tool

TEST MODE

At the top right of the Scene Editor there are four icons that will help you test out your ideas. They include undo, redo, re-centre and test mode.



Test mode lets you quickly see the results of changing properties such as physics values. In the example above, after pressing test the objects react to the gravity of the scene and other objects fall from above:



HOW TO CALL A SCENE FROM YOUR SOURCE CODE

Once you have designed a scene you will need to link to it from your source code. All projects begin with the main.agc file and so it's best to make the link to the scenes from this file. Here's some simple code to illustrate. The name of the example scene is called "gameover". All scenes have .scene added to the end of their name – gameover.scene

```
#include "gameover.scene"        } This line adds the scene
Gameover_setup()                 } Sets up the scene (loads images, makes
                                   sprites etc.)

do
       gameover_sync()           } This function draws the scene
       Print(ScreenFPS())
       Sync()
loop
```

There are some other scene functions you can make use of:

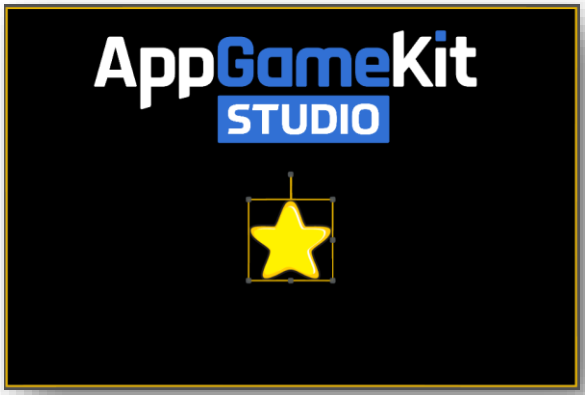
```
Nameofscene_cleanup()            } Call this and all the images and sprites
                                   will be deleted

Nameofscene_fade()               } Sets a fade value to the whole scene
```

This could fade a scene in called gameover:

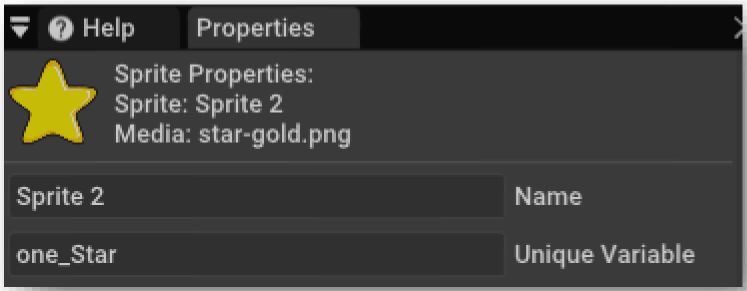
```
//Be sure to call setup before using fade
gameover_setup()
//Fade in the gameover scene
for a = 0 to 100
       gameover_fade(a)
       Sync()
next a
```

LINKING YOUR SOURCE CODE TO THE OBJECTS IN YOUR SCENES

It's all well and good adding images to scenes but how do you control them from your main project? Here's a very simple scene with a star graphic in the middle.



In the properties of the star graphic we have given it the name "Star". AppGameKit Studio will now add the name of the scene to that star and call it "one_Star". So, this is the name of sprite that we will refer to in our code.



Sprite Properties:
Sprite: Sprite 2
Media: star-gold.png

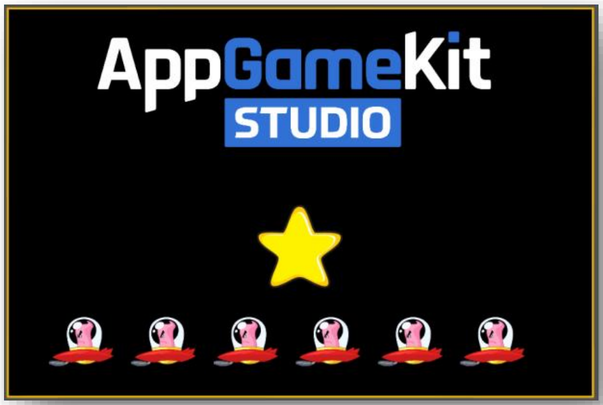
Sprite 2	Name
one_Star	Unique Variable

Let's see the code in main.agc

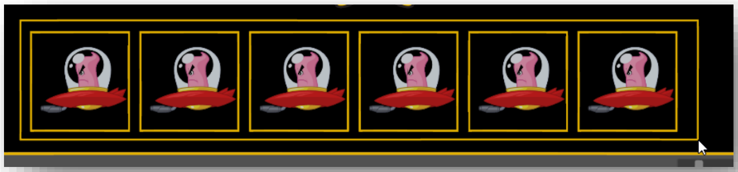
```
#include "one.scene"                } Add the scene to main.agc
one_setup()                         } Set up the scene
rotation=0                          } Create a variable called rotation
do
        SetSpriteAngle(one_Star,rotation)} Change the angle of sprite “one_Star”
        rotation=rotation+1         } Increase the rotation value by 1
        one_sync()                  } Synchronise and draw the frame
     Sync()
loop
```

You can see that it's very easy to make the reference between your code and the objects in the Scene Editor.

What if you want to reference a group of objects? For this you will need to make use of the Array field in the object property window. Let's take the same scene and add a new alien space craft and duplicate it six times. The scene now looks like this:

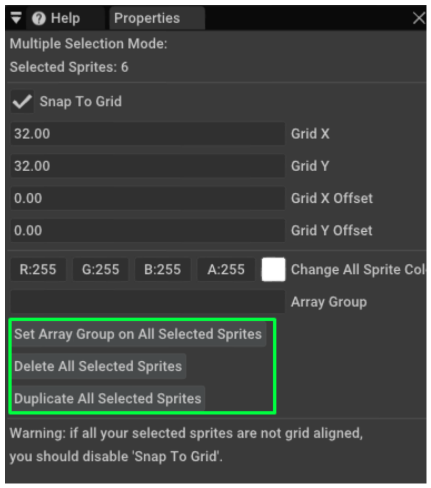


By holding down on the left mouse button, we can drag a rubber band around them all:



The properties window now shows these buttons:

- Set Array Group on All Selected Sprites
- Delete All Selected Sprites
- Duplicate All Selected Sprites



Help | Properties

Multiple Selection Mode:
Selected Sprites: 6

✓ Snap To Grid

32.00	Grid X
32.00	Grid Y
0.00	Grid X Offset
0.00	Grid Y Offset
R:255 G:255 B:255 A:255	Change All Sprite Col
	Array Group

Set Array Group on All Selected Sprites
Delete All Selected Sprites
Duplicate All Selected Sprites

Warning: if all your selected sprites are not grid aligned, you should disable 'Snap To Grid'.

It's the first button we're interested in. In the text field above these buttons we can give the array a name. In this case we'll type "Aliens". AppGameKit Studio will now make an Array called "one_Aliens" – it's an array that deals with the sprites called "Aliens" in the scene called "one".

We'll now use this array reference to move the aliens left and right. Using a loop, we'll cycle through each alien sprite and change their x position on the screen using a value that will be either 1 or -1.

Here’s the new source code to deal with this:

```
#include "one.scene"
one_setup()
rotation=0
counter=1    //counter is either 1 or -1 - adds 1 or takes away 1 from the X
do
       //Rotate the star sprite
       SetSpriteAngle(one_Star,rotation)
       rotation=rotation+1
       //Move the Aliens left and right
       for a=0 to 5
              //This next line uses the Array “on_Aliens” to get the X position
              x=GetSpriteX(one_Aliens[a])
              //Add counter to x
              x=x+counter
              //Set the new X position
              SetSpriteX(one_Aliens[a],x)
       next a

       x=GetSpriteX(one_Aliens[0])
       //Is the first Alien over 300 in the X and are we moving right?
       if x>300 and counter = 1
              counter = -1
       endif
       //Is the X of the first alien less than one and are they moving left?
       if x<1 and counter = -1
              counter = 1
       endif

       //Draw the scene
       one_sync()
    Sync()
loop
```

BEHIND THE SCENES!

For the more inquisitive developer you might want to know about the internal variables and array groups that keep track of the images, sprites and other objects that get added to the scene. All these values and arrays are global values to the whole project. *scenename* is replaced by the name of the scene you are

scenename_fadesprite
 	
scenename _fadeimage as integer
	Used internal to fade scenes. Example code how to fade a scene:
	
	gameover_setup()
	for a = 0 to 100
	  gameover_fade(a)
	  Sync()
	next a

scenename _tween# as float	Used internal to make sure everything moves at the same speed independent of the games's frame rate.
scenename _loaded	Gets set to one when the scene has been loaded and setup.
scenename _images as integer[]	Contains the loaded image IDs
scenename _fonts as integer[]	Holds the loaded font IDs.
scenename _sprites as integer[]	Stores all sprite IDs.
scenename _sprites_count_x as float[] **scenename _sprites_count_y as float[]**	Used internal to move sprites.
scenename _text as integer[]	Contains textbox IDs.
scenename _button as integer[]	Stores the virtual buttons IDs.
scenename _editbox as integer[]	Holds the editbox IDs.

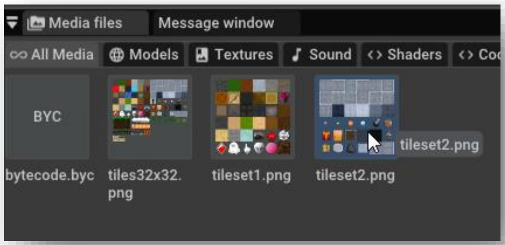
There are four functions you can use with scenes:

- scenename_sync()
- scenename_fade()
- scenename_cleanup()
- scenename_setup()

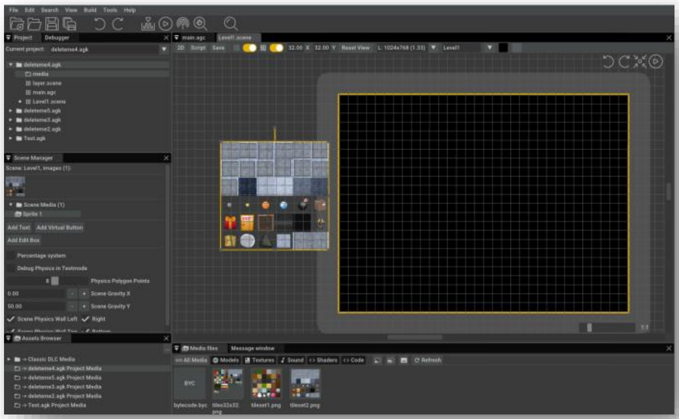
You use "scenename _setup()" to setup the scene and load all its media. If you do not call " scenename_setup()" then " scenename_sync()" will do it for you. For fade to work you must call "scenename_setup()" first.

ADDING A SPRITE SHEET

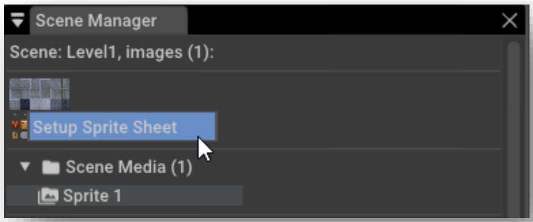
Sprite sheets store a selection of sprites into one image and you can split the sprites they contain into individual sprites using the following procedure. Here we see a sprite sheet image that has 6x6 tiles mapped onto it.



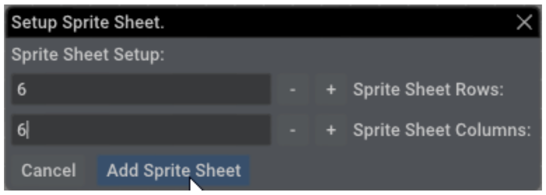
We need to drag the image into the scene but we don't want it to be seen so we place it into the invisible area of the scene (and it can be deleted from view too if you prefer). A small thumbnail will show up in the Scene Manager window to indicate its now part of the scene.



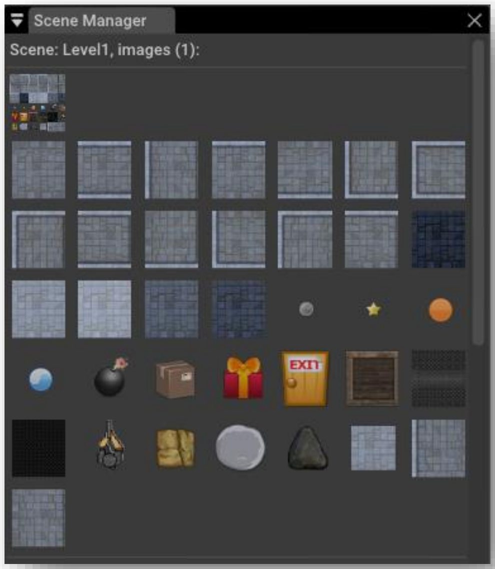
Right click on the image and a small menu option will appear "Setup Sprite Sheet":



The next dialogue box will appear. It needs to know how many sprites there are across and down the gird of sprites. In this case it's 6x6 which means there are 36 sprites:



When you are sure you have the correct row and column counts, click on "Add Sprite Sheet". AppGameKit Studio will now cut up each image and make the sprites, these will now be shown in the Scene Manager like this:



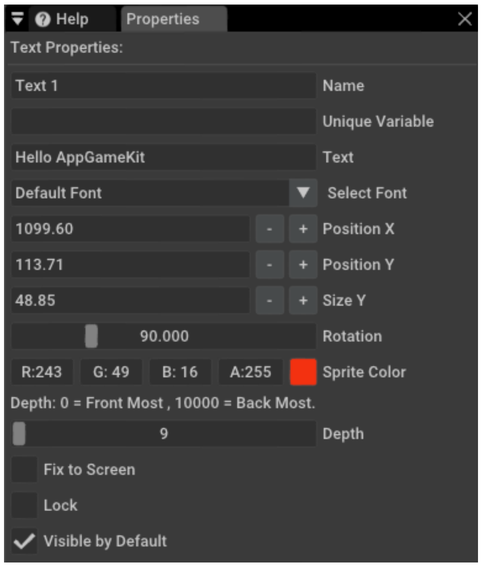
The sprites can now be dragging into your scene and used in Draw Mode as you like.

TEXT OBJECTS

Text objects are easy to add to a scene, just click the *Add Text* button in the Scene Manager and a new text object will be created and appear in the top left of the scene. Many of the properties of text objects are similar to the Sprite Properties but there are some specific fields relevant only to texts.

Text Properties

Name	Just a reference name for use during editing. You don't need to set it or make it unique.
Unique Variable	A global variable to use with this text object so you can reference it from your code.
Text	The text you want your text object to show on screen.
Select Font	If you have added TTF fonts to your scene, you will be able to select which font your text object will use.
Position X / Y	The X and Y position of the text object (drawn from the top left of the text object).
Size	The size of the text object in pixels.
Rotation	The rotation of the text object, for example to have text down the screen set this to 90.
Sprite Color	The colour of the text object.
Depth	Set the draw depth of the text object. For example, you might want text for a score display to always be in front of all game sprites so you would set the depth to a low value.
Fix to Screen	This stops the text from moving if the screen is made to scroll.
Lock	Locks the text object to ensure its not accidentally moved.
Visible by Default	If ticked, the text will show when the scene is displayed.



Text Properties:

Text 1	Name
	Unique Variable
Hello AppGameKit	Text
Default Font	Select Font
1099.60	Position X
113.71	Position Y
48.85	Size Y
90.000	Rotation
R:243 G: 49 B: 16 A:255	Sprite Color

Depth: 0 = Front Most , 10000 = Back Most.

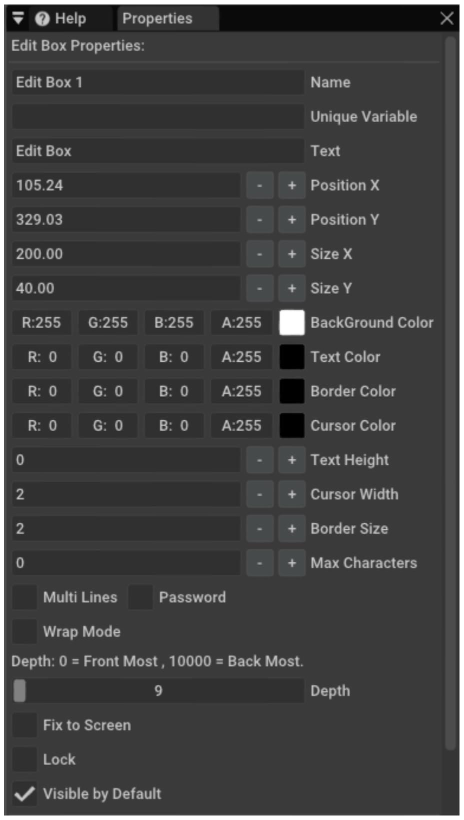
9	Depth
	Fix to Screen
	Lock
✓	Visible by Default

EDIT BOX OBJECTS

Edit boxes are an easy way to add a text entry fields into your scenes. They are used by apps to allow users to enter text strings. Things like the players name for a high score table.

Edit Box Properties

Name	Just a reference name for use during editing. You don't need to set it or make it unique.
Unique Variable	A global variable to use with this text object so you can reference it from your code.
Text	The default text you want to appear in the edit box when it is first shown.
Position X/Y	The top left X and Y position of the Edit box.
Size X/Y	The size of the Edit box in pixels.
Background Color	The color of the background (behind the text).
Text Color	The color of the text.
Border Color	The border color of the Edit box.
Cursor Color	The color of the flashing cursor.
Text Height	Sets the height in world coordinates of the text in the Edit box. Set to 0 to use the default which is the Edit box's height minus two.
Cursor Width	The width of the Edit box.
Border Size	The size of the surrounding border.
Max Characters	Sets the maximum number of characters that can be entered in this Edit box. Use 0 for unlimited.
Multi Lines	Sets whether the Edit box will wrap text to a new line when it reaches the edge of the Edit box.
Password	Sets whether the Edit box displays stars instead of the text input.
Wrap Mode	When multi lines is not used, this option sets whether the single line of text will scroll to the right or wrap to a new line when it over flows the Edit box width.
Depth	Set the draw depth of the Edit box.
Fix to Screen	This stops the Edit box from moving if the screen scrolls.
Lock	Locks the Edit box object to ensure its not accidentally moved.
Visible by Default	If ticked, the Edit box will show when the scene is displayed.



Edit Box Properties:

Edit Box 1	Name
	Unique Variable
Edit Box	Text
105.24	Position X
329.03	Position Y
200.00	Size X
40.00	Size Y
R:255 G:255 B:255 A:255	BackGround Color
R: 0 G: 0 B: 0 A:255	Text Color
R: 0 G: 0 B: 0 A:255	Border Color
R: 0 G: 0 B: 0 A:255	Cursor Color
0	Text Height
2	Cursor Width
2	Border Size
0	Max Characters

Multi Lines
Password
Wrap Mode

Depth: 0 = Front Most , 10000 = Back Most.

9 Depth

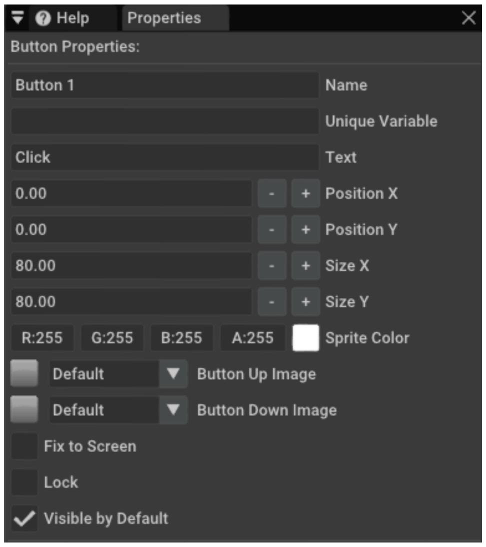
Fix to Screen
Lock
✓ Visible by Default

VIRTUAL BUTTON OBJECTS

Virtual Buttons are a quick way to add user input gadgets into your app scenes. The buttons have default grey up & down images and you can change these to images of your own design.

Button Properties

Name	Just a reference name for use during editing. You don't need to set it or make it unique.
Unique Variable	A global variable to use with this button object so you can reference it from your code.
Text	The text you want to show in the button.
Position X/Y	The X and Y position of the Virtual Button in the scene.
Size X/Y	The size of the Virtual Button.
Button Color	The colour of the button.
Button Up Image	The image to use when the button is not being clicked – button up state.
Button Down Image	When clicked, this image will be shown – button down state.
Fix to Screen	This stops the Virtual Button from moving if the screen scrolls.
Lock	Locks the Virtual Button to ensure its not accidentally moved.
Visible by Default	If ticked, the Virtual Button will show when the scene is displayed.

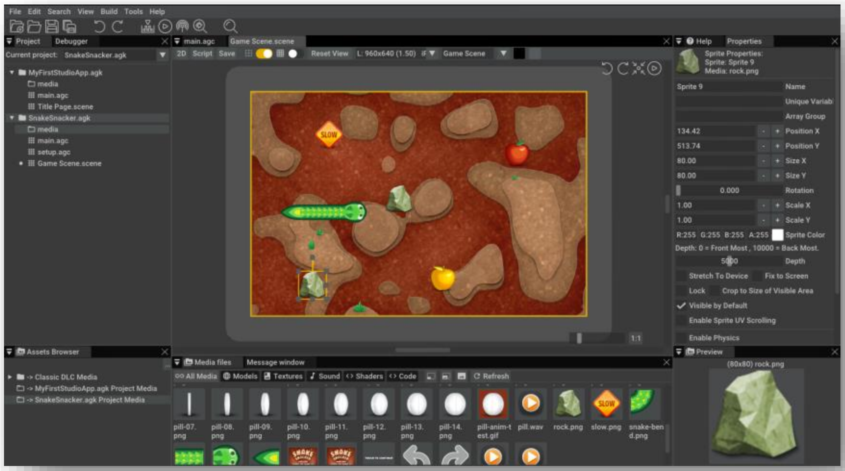


Button Properties:

Button 1	Name
	Unique Variable
Click	Text
0.00	Position X
0.00	Position Y
80.00	Size X
80.00	Size Y
R:255 G:255 B:255 A:255	Sprite Color
Default	Button Up Image
Default	Button Down Image
	Fix to Screen
	Lock
✓	Visible by Default

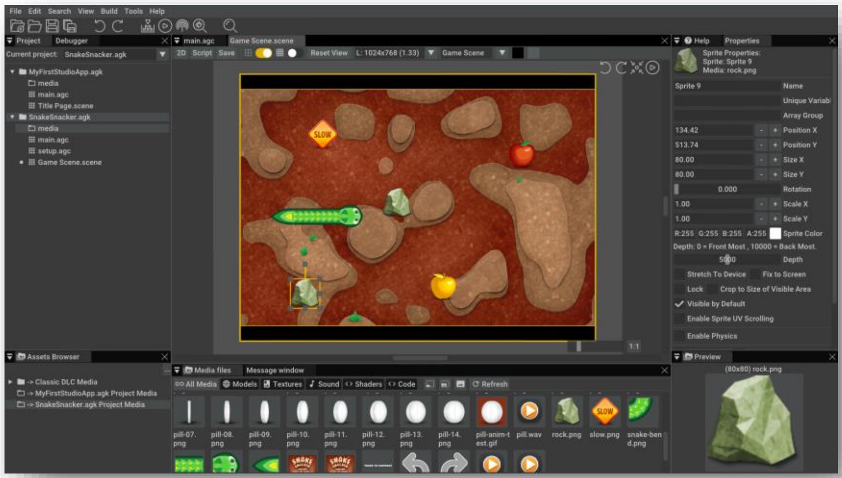
UNDERSTANDING THE BASE RESOLUTION

When you start a new scene a default resolution and orientation is set. This resolution acts as the base for all your other resolutions. The idea is for you to pick a resolution to be the base from which others will rescale and reposition from. In this example we'll choose Landscape 960 x 640 as our base resolution. You first set out the entities into the scenes using this base resolution:



The aspect ratio of 960x640 is 1.5 (960/640). When you switch to view another resolution (for example 1024x768) all of the entities will be rescaled and repositioned whilst maintaining the original aspect ratio.

Because the aspect ratio for this resolution is 1.33 (1024/768= 1.33), the background cannot fill the full screen and as a result empty borders now show.

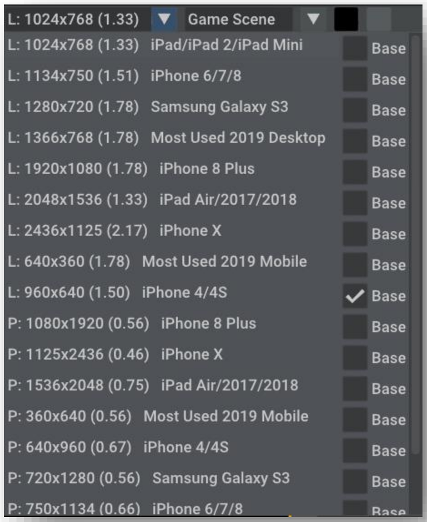


The way to fix this is to expand the size of the background to ensure it takes into account these border areas. You should check all the resolutions in the drop-down menu to ensure your app will display its best on any device aspect ratio. You can also enter custom resolutions.

The tick on the right indicated the base resolution that has been chosen for this scene.

The L: and P: is short for Landscape and Portrait orientations.

Testing your app on different devices that have different resolutions is an important aspect to ensuring you create a robust app that will work for any user. As devices develop larger resolutions will become available and so you'll need to update your app if it needs to be changed to work for them.



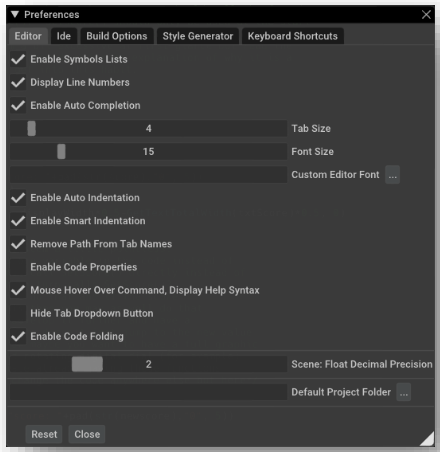
L: 1024x768 (1.33)	iPad/iPad 2/iPad Mini	Base
L: 1134x750 (1.51)	iPhone 6/7/8	Base
L: 1280x720 (1.78)	Samsung Galaxy S3	Base
L: 1366x768 (1.78)	Most Used 2019 Desktop	Base
L: 1920x1080 (1.78)	iPhone 8 Plus	Base
L: 2048x1536 (1.33)	iPad Air/2017/2018	Base
L: 2436x1125 (2.17)	iPhone X	Base
L: 640x360 (1.78)	Most Used 2019 Mobile	Base
L: 960x640 (1.50)	iPhone 4/4S	✓ Base
P: 1080x1920 (0.56)	iPhone 8 Plus	Base
P: 1125x2436 (0.46)	iPhone X	Base
P: 1536x2048 (0.75)	iPad Air/2017/2018	Base
P: 360x640 (0.56)	Most Used 2019 Mobile	Base
P: 640x960 (0.67)	iPhone 4/4S	Base
P: 720x1280 (0.56)	Samsung Galaxy S3	Base
P: 750x1134 (0.66)	iPhone 6/7/8	Base

PREFERENCES

You can access the Properties window for AppGameKit Studio in the Edit drop-down menu. The various settings are all described here.

EDITOR TAB

These preferences focus on various editor settings.



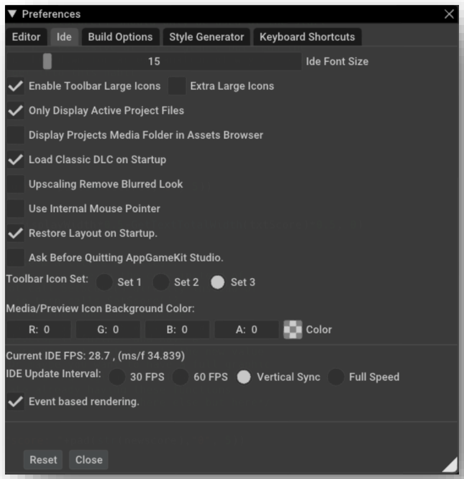
- ✓ Enable Symbols Lists
- ✓ Display Line Numbers
- ✓ Enable Auto Completion
- 4 Tab Size
- 15 Font Size
- Custom Editor Font ...
- ✓ Enable Auto Indentation
- ✓ Enable Smart Indentation
- ✓ Remove Path From Tab Names
- Enable Code Properties
- ✓ Mouse Hover Over Command, Display Help Syntax
- Hide Tab Dropdown Button
- ✓ Enable Code Folding
- 2 Scene: Float Decimal Precision
- Default Project Folder ...

Enable Symbols Lists	This turns the symbols list on and off in the IDE (it shows a drop-down list of functions and global variables).
Display Line Numbers	Toggles the display of line numbers in the IDE.
Enable Auto Completion	Suggested commands will be listed as you type into the IDE. If you don't like this then you can turn it off here.
Tab Size	The size of indentation tabs within your code can be defined here. You will see code change as you move the tab too.
Font Size	The font size of the code in the IDE.
Custom Editor Font	Change the IDE font to one of your own choice. You just need to have a fixed width ttf font available.
Enable Auto Indentation	This keeps the current indentation as you press enter for a new line. Without it on you are returned to the start of the new line.
Enable Smart Indentation	This option checks what you write in the line when you press enter and adds an indentation based on that. Let's say you type words like If, For, While, Do - it will add additional indentation. If it finds stop words like "Then" it will not perform any additional indentation.
Remove Path from Tab Names	The folder paths for the source file tabs will not be shown if this is ticked, tidying up the display of the list. This only applies to files that are placed outside the project or files placed in sub folders of the project.
Enable Code Properties	Turns on the code properties feature. See the section on Code Properties for more detailed information.
Mouse Hover Over Command, Display Help Syntax	When this is ticked you will see a help hint appear near the mouse pointer when you hover over a command in your code. It's ideal for reminding you of the necessary parameters of a command.

Hide Tab Dropdown Button	Each window has a small tab drop down menu. By ticking this option you can hide these menu buttons given you a cleaner display.
Enable Code Folding	When you're coding a large project, you might want to hide some parts of your code that you know are complete and work well. This feature lets you hide such code. See the section about code folding for a more detailed description.
Scene: Float Decimal Precision	This allows you to control how many decimal places show in the values shown in the Scene Editor.
Default Project Folder	Select the folder where you want to create new projects in.

IDE TAB

These properties deal with how the IDE looks and runs.



Ide Font Size: 15

✓ Enable Toolbar Large Icons ☐ Extra Large Icons

✓ Only Display Active Project Files

☐ Display Projects Media Folder in Assets Browser

✓ Load Classic DLC on Startup

☐ Upscaling Remove Blurred Look

☐ Use Internal Mouse Pointer

✓ Restore Layout on Startup.

☐ Ask Before Quitting AppGameKit Studio.

Toolbar Icon Set: Set 1 Set 2 [✓ Set 3]

Media/Preview Icon Background Color:

R: 0 G: 0 B: 0 A: 0 Color

Current IDE FPS: 28.7 , (ms/f 34.839)

IDE Update Interval: 30 FPS 60 FPS [✓ Vertical Sync] Full Speed

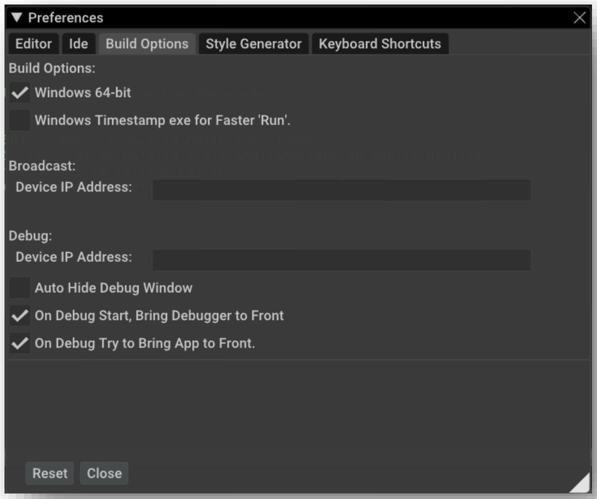
✓ Event based rendering.

IDE Font Size	With this you can change the font used throughout the AppGameKit Studio editor.
Enable Toolbar Large Icons / Extra Large Icons	There are three sizes for the tool bar and you can control it with these settings. If these two are off you see the smallest and if you tick any one of these you will see the other sizes.
Only Display Active Project Files	This is on by default. If you tick it off then all the files from the projects list will be displayed at the top of the IDE in tabs.
Display Projects Media Folder in Assets Browser	When this is ticked, any loaded projects media folders will be displayed in the Asset Browser for quick access of media.
Load Classic DLC on Startup	If you own Downloadable Content from AppGameKit Classic then you can show their folders in the Asset Browser for convenience.
Upscaling Remove Blurred Look	Use this option if you prefer a sharper rendering of the code font.
Use Internal Mouse Pointer	On some desktop systems with dual monitor setups and using the full screen mode, one of the monitors might not have a mouse pointer. We have added this special internal mouse pointer if you experience this.
Restore Layout on Startup	If you move around the various windows in the UI and want these new positions to be remembered then make sure this is ticked.
Ask Before Quitting AppGameKit Studio	?????????????

Toolbar Icon Set	Choose between three different icon types for the tool bar.
Media/Preview Icon Background Color	Change the background used by the preview window. You would do this if the media being viewed clashes with the background.
Current IDE FPS	Shows a live update of the current frame rate of the IDE.
IDE Update Interval	You can set how often the IDE updates, 30 FPS, 60 FPS, in sync with the vertical sync or at full speed.
Event based rendering	Tick this and it will skip rendering frames if there are no events, it will do this no matter what FPS you set the IDE to update at. Events can be keyboard or mouse movements, moving windows – any event that would require that the full window needs to be updated and re-rendered. On lower spec devices this helps when you are running the editor and your own app.

BUILD OPTIONS

When you compile your project, the source code gets changed into bytecode and this is run and interpreted by a player program (*projectname.exe* on Windows, *projectname.app* on Mac and *LinuxPlayer64* on Linux). The final executable program can be in 32 or 64 bit formats on Windows and Linux, Mac only supports 64 bit.



Build Options:
✓ Windows 64-bit
Windows Timestamp exe for Faster 'Run'.

Broadcast:
Device IP Address:

Debug:
Device IP Address:
Auto Hide Debug Window
✓ On Debug Start, Bring Debugger to Front
✓ On Debug Try to Bring App to Front.

Build Options

Windows 64-bit	On Windows and Linux, you can choose to compile as a 32 bit or 64 bit executable. Mac can only compile to 64 bit.
Windows Timestamp exe for Faster 'Run'	If this is ticked, the exe file will not be updated every compile, it saves time and lets you run your tests quicker.

Debug

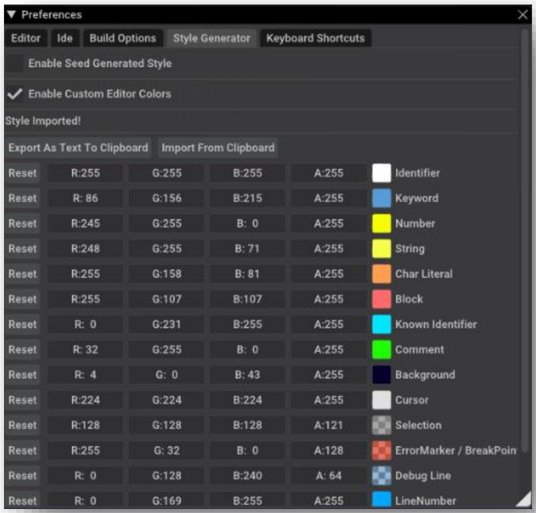
Device IP Address	If you want to debug across devices then you can use this field to enter the IP address of the other device. For example, you might develop on a PC and want to run the app on another device so you can see the code on one screen and the app on another.

Auto Hide Debug Window	If this is ticked then the debugger window will only appear when you start a debugging session (when you need it).
On Debug Start, Bring Debugger to Front	When starting a debug session and this is ticked, the debugger window will show in front of any other window in the area of UI where it's located.
On Debug Try to Bring App to Front	If selected, when you start a debug session the project app you are running will be forced to appear above the main AppGameKit Studio user interface.

For full details about the debugger see the Debugger section of this user guide.

STYLE GENERATOR

As a game developer you'll be spending a lot of time writing programs in the IDE. It's important you're happy with the colours used in the IDE. You can choose from the various default color setups from the *View/Choose Color Scheme* drop-down menu. If these don't work for you then you can design your own with either a random generator or a custom editor.



Enable Seed Generated Style

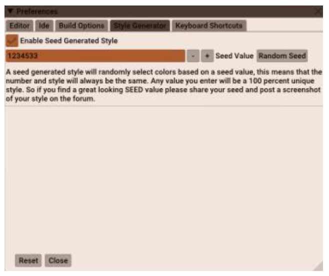
✓ Enable Custom Editor Colors

Style Imported!

Export As Text To Clipboard | Import From Clipboard

Reset	R:255	G:255	B:255	A:255	Identifier
Reset	R: 86	G:156	B:215	A:255	Keyword
Reset	R:245	G:255	B: 0	A:255	Number
Reset	R:248	G:255	B: 71	A:255	String
Reset	R:255	G:158	B: 81	A:255	Char Literal
Reset	R:255	G:107	B:107	A:255	Block
Reset	R: 0	G:231	B:255	A:255	Known Identifier
Reset	R: 32	G:255	B: 0	A:255	Comment
Reset	R: 4	G: 0	B: 43	A:255	Background
Reset	R:224	G:224	B:224	A:255	Cursor
Reset	R:128	G:128	B:128	A:121	Selection
Reset	R:255	G: 32	B: 0	A:128	ErrorMarker / BreakPoin
Reset	R: 0	G:128	B:240	A: 64	Debug Line
Reset	R: 0	G:169	B:255	A:255	LineNumber

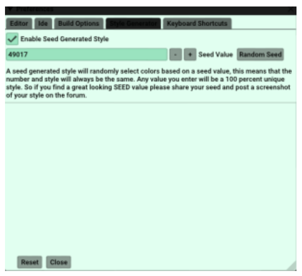
If you're stuck for number ideas just hit the Random Seed button and a number will be generated. Hopefully you will discover a cool new IDE look – here are some we created earlier:



✓ Enable Seed Generated Style

1234533

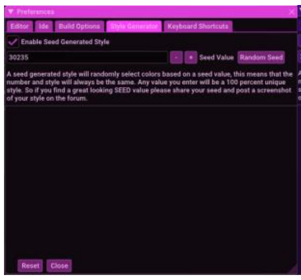
A seed generated style will randomly select colors based on a seed value, this means that the number and style will always be the same. Any value you enter will be a 100 percent unique style. So if you find a great looking SEED value please share your seed and post a screenshot of your style on the forum.



✓ Enable Seed Generated Style

49017

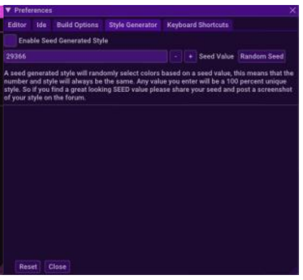
A seed generated style will randomly select colors based on a seed value, this means that the number and style will always be the same. Any value you enter will be a 100 percent unique style. So if you find a great looking SEED value please share your seed and post a screenshot of your style on the forum.



✓ Enable Seed Generated Style

30235

A seed generated style will randomly select colors based on a seed value, this means that the number and style will always be the same. Any value you enter will be a 100 percent unique style. So if you find a great looking SEED value please share your seed and post a screenshot of your style on the forum.



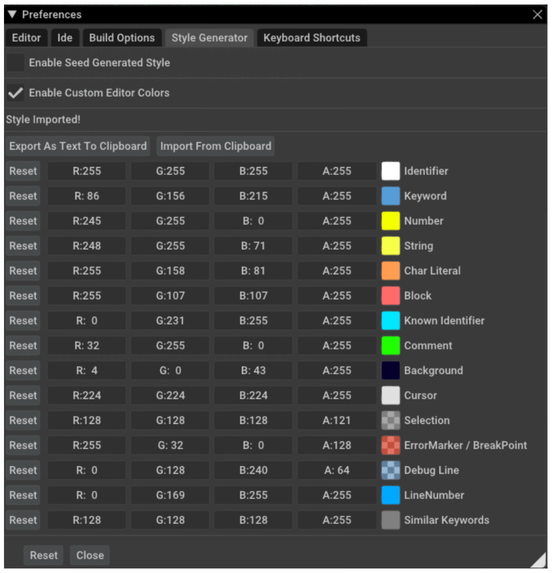
Enable Seed Generated Style

29366

A seed generated style will randomly select colors based on a seed value, this means that the number and style will always be the same. Any value you enter will be a 100 percent unique style. So if you find a great looking SEED value please share your seed and post a screenshot of your style on the forum.

While the random generator is a lot of fun it’s unlikely to land you on the perfect design. The custom editor option lets you set every colour type of the IDE. Once ticked, a list of items that are colored is shown. To change any of these colors just click on the color you want to modify and a colour box will appear. You can change the color and see the change instantly in the IDE behind the preferences window.

There are two buttons to export and import color scheme designs. If you come up with a great color palette please share it with others in the AppGameKit community.



☐ Enable Seed Generated Style

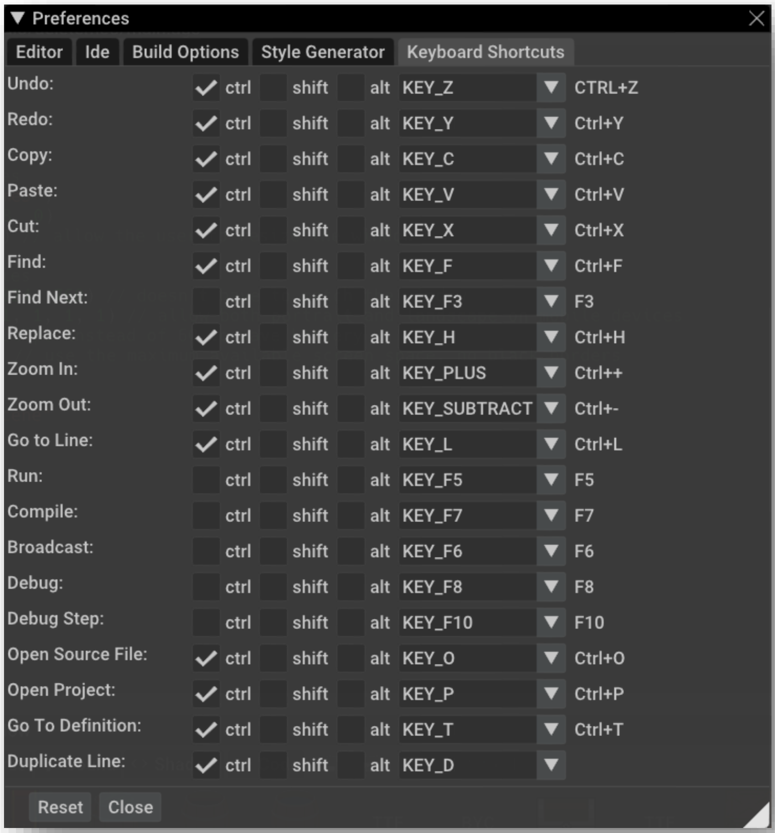
✓ Enable Custom Editor Colors

Style Imported!

Reset	R:255	G:255	B:255	A:255	⬜ Identifier
Reset	R: 86	G:156	B:215	A:255	🟦 Keyword
Reset	R:245	G:255	B: 0	A:255	🟨 Number
Reset	R:248	G:255	B: 71	A:255	🟨 String
Reset	R:255	G:158	B: 81	A:255	🟧 Char Literal
Reset	R:255	G:107	B:107	A:255	🟥 Block
Reset	R: 0	G:231	B:255	A:255	🟦 Known Identifier
Reset	R: 32	G:255	B: 0	A:255	🟩 Comment
Reset	R: 4	G: 0	B: 43	A:255	⬛ Background
Reset	R:224	G:224	B:224	A:255	⬜ Cursor
Reset	R:128	G:128	B:128	A:121	▨ Selection
Reset	R:255	G: 32	B: 0	A:128	▨ ErrorMarker / BreakPoint
Reset	R: 0	G:128	B:240	A: 64	▨ Debug Line
Reset	R: 0	G:169	B:255	A:255	🟦 LineNumber
Reset	R:128	G:128	B:128	A:255	⬜ Similar Keywords

KEYBOARD SHORTCUTS

In this tab you can remap many of the features that have quick key choices. Simply choose the key combination you prefer and the editor will follow those rules from then on.



Undo:	✓ ctrl	shift	alt	KEY_Z	CTRL+Z
Redo:	✓ ctrl	shift	alt	KEY_Y	Ctrl+Y
Copy:	✓ ctrl	shift	alt	KEY_C	Ctrl+C
Paste:	✓ ctrl	shift	alt	KEY_V	Ctrl+V
Cut:	✓ ctrl	shift	alt	KEY_X	Ctrl+X
Find:	✓ ctrl	shift	alt	KEY_F	Ctrl+F
Find Next:	ctrl	shift	alt	KEY_F3	F3
Replace:	✓ ctrl	shift	alt	KEY_H	Ctrl+H
Zoom In:	✓ ctrl	shift	alt	KEY_PLUS	Ctrl++
Zoom Out:	✓ ctrl	shift	alt	KEY_SUBTRACT	Ctrl+-
Go to Line:	✓ ctrl	shift	alt	KEY_L	Ctrl+L
Run:	ctrl	shift	alt	KEY_F5	F5
Compile:	ctrl	shift	alt	KEY_F7	F7
Broadcast:	ctrl	shift	alt	KEY_F6	F6
Debug:	ctrl	shift	alt	KEY_F8	F8
Debug Step:	ctrl	shift	alt	KEY_F10	F10
Open Source File:	✓ ctrl	shift	alt	KEY_O	Ctrl+O
Open Project:	✓ ctrl	shift	alt	KEY_P	Ctrl+P
Go To Definition:	✓ ctrl	shift	alt	KEY_T	Ctrl+T
Duplicate Line:	✓ ctrl	shift	alt	KEY_D	

CODE PROPERTIES

To improve the visibility of key data in your code you can expose such data so they appear as UI gadgets in the editor. They can then be changed while your app is running and you can see the immediate affect such changes have. These values will be sent to and from the IDE and the running app.

New Code Properties System

Following feedback after the initial launch we have improved how the code properties are setup. You can now use the new #export keyword to indicate the use of a code property. These uses of #export are for setting up information about the values that can be edited:

`#export(header,"My Cool Project" )`	Creates the header text “My Cool Project”
`#export (message,"Change the values below")`	Makes the message “Change the values below”
`#export (separator)`	Displays a line separator

Here are some examples of how you can setup Global variables and at the same time create editable properties for them:

`global mystring$ = "Level Name" #export ( string , "Change level name" )`	The level’s name is exported as a code property string that you can now edit.
`global fogmin = 255 #export (integer,"Set fog minimum range" )`	An integer value for a fog setting.
`global bounce# = 300.000000 #export (float,"Bounce Car.")`	A floating point value for setting how bouncy a car is in a demo.
`global myobject$ = "SM_Veh_Car_Police_01.fbx" #export (selectfile,"Select 3D object file.")`	The file name used when loading a 3D object is exposed. Thus allowing you to easily select another file each time you compile the project.

Here’s some code that shows each of these code properties being setup in source and below the code to the right are the resulting gadgets that are created.

```
#export(header,"My Cool Project" )
#export (message,"Change the values below")
#export (separator)
global mystring$ = "Level Name" #export ( string , "Change level name" )
global fogmin = 255 #export (integer,"Set fog minimum range" )
global bounce# = 300.000000 #export (float,"Bounce Car.")
global myobject$ = "SM_Veh_Car_Police_01.fbx" #export (selectfile,"Select 3D object file.")
```

Older Code Properties System

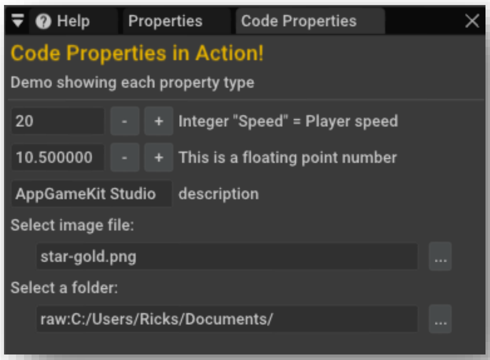
This was our first implementation and it can still be used. We've left it in to ensure user projects are kept compatible.

A special keyword is used in the IDE to initiate the use of code properties. It starts with the comment characters //, then an open square bracket **[**, is then followed by **IDE**, **GUI** and **ADD**, then close square bracket **]**, and finally a comma **,** so the keyword is: **//[IDEGUIADD],**

Once that special keyword is typed in, the IDE will recognise it as the start of a Code Property line. You can then add other parameters after the comma, here's a list:

`//[IDEGUIADD],header,`*`textstring`*	Creates a header text
`//[IDEGUIADD],message,`*`textstring`*	Shows a text description message
`//[IDEGUIADD],separator,`	Draws a line separator
`//[IDEGUIADD],integer,description`	Edit box for an integer value
`//[IDEGUIADD],string,description`	Edit box for a text string
`//[IDEGUIADD],selectfile`	A file selector for choosing a file
`//[IDEGUIADD],selectfolder`	A file selector for choosing a folder

Here's some code that shows each of these code properties being setup in source and below the code to the right are the resulting gadgets that are created.



Code Properties in Action!
Demo showing each property type
20 - + Integer "Speed" = Player speed
10.500000 - + This is a floating point number
AppGameKit Studio description
Select image file:
star-gold.png ...
Select a folder:
raw:C:/Users/Ricks/Documents/ ...

```
//[IDEGUIADD],header,Code Properties in Action!
//[IDEGUIADD],message,Demo showing each
property type
//[IDEGUIADD],separator,
global speed=20 //[IDEGUIADD],integer,Integer
"Speed" = Player speed
global float#=10.500000
//[IDEGUIADD],float,This is a floating point
number
global bal name$ = "AppGameKit Studio"
//[IDEGUIADD],string,description
global my_image$ = "star-gold.png" //[IDEGUIADD],selectfile,Select image file
global my_folder$ = "raw:C:/Users/Ricks/Documents/"//[IDEGUIADD],selectfolder,Select a
folder
```

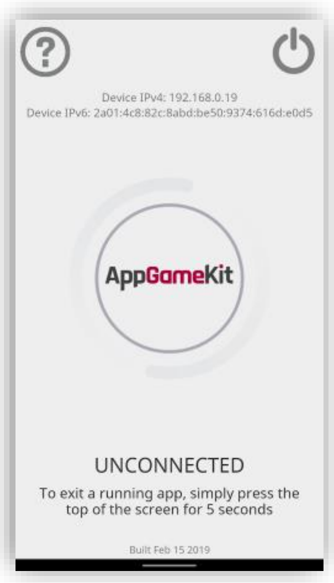
Once the code has these keywords setup, you can change the values using the controls in the Code Properties window and the source code will update with these changes. What's really cool is that you can run your app in Debug mode and as your app is running you can change the values and see the changes happen in your live running app!

BROADCASTING

The broadcasting feature of AppGameKit Studio helps you to test your creations on real mobile devices without needing to export your projects. It's a great time saver and lets you try your results on device which is usually very different to how you would use the same app on a desktop or laptop device.

There are a few things you need to setup before you can make use of broadcasting:

- First you obviously have to have at least one mobile device (iOS or Android based)
- Your mobile device needs to be on the same Wi-Fi channel as your desktop/laptop that's running AppGameKit Studio
- The AppGameKit Player app must be running on the mobile device.



Device IPv4: 192.168.0.19
Device IPv6: 2a01:4c8:82c:8abd:be50:9374:616d:e0d5

UNCONNECTED

To exit a running app, simply press the top of the screen for 5 seconds

Built Feb 15 2019

Once all of the above is setup, you can load up your project into AppGameKit Studio and then you can press the Broadcast button in the toolbar.

The project file will be transmitted from the desktop device over the Wi-Fi to the AppGameKit Player app on the mobile device. When the transmission is 100% complete the project will be run on the mobile device. Each time you broadcast only the changes will be sent over and thus reduce the time to broadcast.

You can end the broadcast by two methods:

- Press the broadcast icon again on the desktop
- Hold down on the top right-hand corner of the running app for about 5 seconds

The app shows some device IP values and these are helpful for broadcasting if the player app doesn't automatically connect with the IDE. You can enter the IP details into the IDE to make it connect directly.

DEBUGGER

At times you will find that your intended code is not doing what you expected and no matter how many times you think through the logic you cannot fathom why the results differ. At such times you need the help of a Debugging tool and the good news is that AppGameKit Studio has a great Debugger!

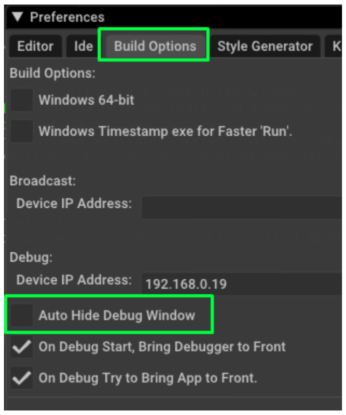
The Debugger is called up by pressing this icon in the toolbar or via the Build drop-down menu. When you call the Debugger, it will run your app in a special way – the AppGameKit Studio editor will be communicating directly with your running project, receiving data such as variable and array data, and it can also send back data to change the running app.

THE DEBUG WINDOW

The debug window will only appear when you start the debug process. There is an option in the *Preferences* that allows you to have the Debug windows always visible if you prefer.

To always show it, open *Preferences*, choose the *Build Options* tab and make sure *Auto Hide Debug Window* is un-ticked.

It's useful to have the debug window available when you are coding because you might want to add a variable or an array to the watch list. If the debug window is visible you can see you have successfully added it and will be sure it will show when you start your debug session.



Build Options:

Windows 64-bit

Windows Timestamp exe for Faster 'Run'.

Broadcast:

Device IP Address:

Debug:

Device IP Address: 192.168.0.19

Auto Hide Debug Window

✓ On Debug Start, Bring Debugger to Front

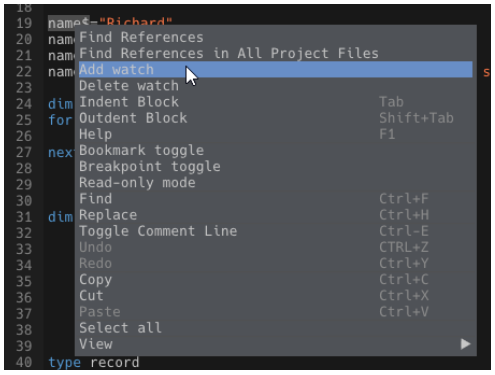
✓ On Debug Try to Bring App to Front.

CREATING A WATCH LIST

When using the Debugger, you will need to prepare some watch values. These are the variables and arrays that you expect are not being set with the desired results in your app and so you need to watch what values they change to during the running of the app. There are two ways to add data to the watch list.

The first is directly from your project code. In the IDE, highlight the variable you want to watch and then right click on it, from the menu that appears select *Add watch*. The variable will now be shown in the watch list in the Debugger Window.

The second way is to just type the variable you want to watch into field in the Debugger Window. The next section details all the features of this control area.

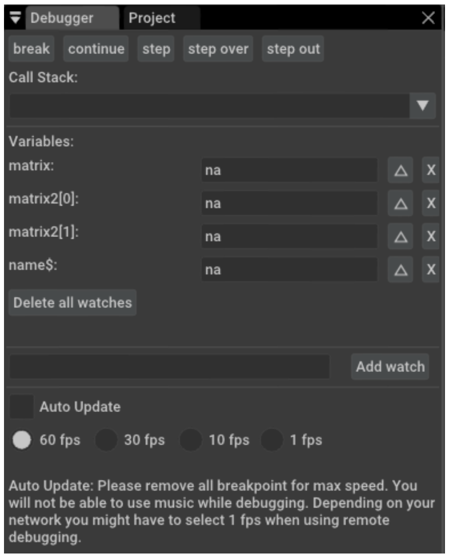


```
name$="Richard"
nam
nam
nam

dim
for

nex



dim








type record
```

THE DEBUGGER WINDOW

This is the main Debugger area where you can watch and set data, and where you can control the debugging process.

Break, continue, step, step over, step out	Use these buttons to break the running of your app and then you can either continue it or single step through code. Step over will ignore the next command and step out will jump back out of a function (useful if you know that code works fine and you're dealing with code higher up the call logic).
Call Stack	The call stack will display where you currently are in the debugger and what function was used to reach that point.
Variables	This is a list of the watched variables for the purpose of finding and fixing your particular issue. Press the triangle icon next to the variable to set a new value for the variable. Click the X to remove the variable from the list.
Delete all watchers	A quick way to remove all the watched variables from the list.
Add watch	Type in the name of a variable, click *Add watch* and it will be listed above.
Auto Update	Tick this if you want to actively watch variables change at the same time as your app is running.
60/30/10/1 fps	Set the frame rate as to how often the debugger shows the updated variables.



Variable	Value
matrix:	na
matrix2[0]:	na
matrix2[1]:	na
name$:	na

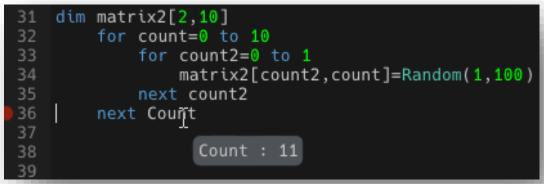
Auto Update: Please remove all breakpoint for max speed. You will not be able to use music while debugging. Depending on your network you might have to select 1 fps when using remote debugging.

BREAKPOINTS

A breakpoint is a location in your code where you want to stop the app in its tracks. As soon as the debugger reaches this location the app will stop and you can take over control of what happens next and view the state of any values.

To set a breakpoint, just click left of the line number column and a red dot will mark this as the point of a breakpoint. A right click on a line will also show a menu option allowing you to do the same.

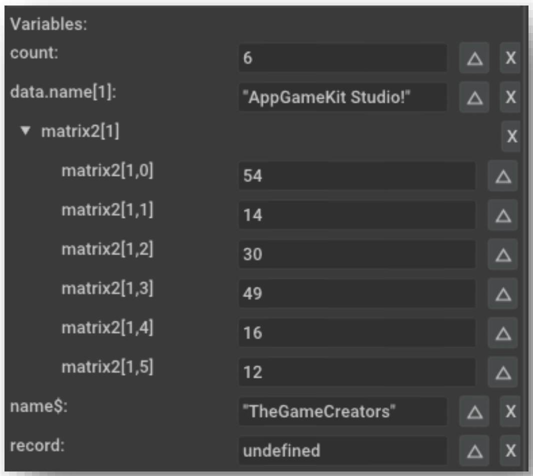
You might want to review the values that are showing in the watch list or you might want to start stepping through the next part of the code, carefully reviewing what changes are made and how the code flows. You can also hover over variables in the IDE code and reveal the values they have been set to.



```
dim matrix2[2,10]
    for count=0 to 10
        for count2=0 to 1
            matrix2[count2,count]=Random(1,100)
        next count2
    next Count
```

VARIABLE TYPES

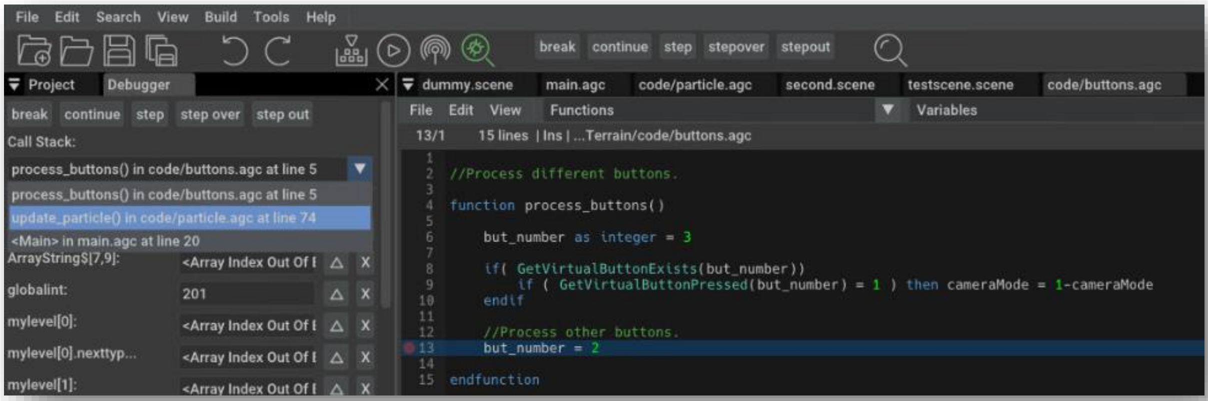
You can add global variables, stings, arrays and types to the watch list. Here's an example of a range of different data types being watched:



```
Variables:
count:              6
data.name[1]:       "AppGameKit Studio!"
▼ matrix2[1]
    matrix2[1,0]    54
    matrix2[1,1]    14
    matrix2[1,2]    30
    matrix2[1,3]    49
    matrix2[1,4]    16
    matrix2[1,5]    12
name$:              "TheGameCreators"
record:             undefined
```

THE CALL STACK

The call stack will display where you currently are in the code and what functions were used to react that point. As you can see in this example, the breakpoint is in the function *process_buttons()* , so this is displayed at the top. *process_buttons()* was called from the function *update_particle()*, so this is entry number 2 in the list. Finally, *update_particle()* was called from *<Main>*, the main loop at line 20, so this is display last in the list.


File Edit Search View Build Tools Help
break continue step stepover stepout
Project Debugger
dummy.scene main.agc code/particle.agc second.scene testscene.scene code/buttons.agc
break continue step step over step out
File Edit View Functions Variables

```
Call Stack:
process_buttons() in code/buttons.agc at line 5
process_buttons() in code/buttons.agc at line 5
update_particle() in code/particle.agc at line 74
<Main> in main.agc at line 20
ArrayString$[7,9]:    <Array Index Out Of I
globalint:            201
mylevel[0]:           <Array Index Out Of I
mylevel[0].nexttyp... <Array Index Out Of I
mylevel[1]:           <Array Index Out Of I
```

```
13/1    15 lines  | Ins | ...Terrain/code/buttons.agc

//Process different buttons.

function process_buttons()

    but_number as integer = 3

    if( GetVirtualButtonExists(but_number))
        if ( GetVirtualButtonPressed(but_number) = 1 ) then cameraMode = 1-cameraMode
    endif

    //Process other buttons.
    but_number = 2

endfunction
```

EXPORTING YOUR PROJECTS

When you are ready to publish your app you will need to first export in a format that supports the Google Play Store, Amazon App Store and iOS App Store. Exporting APK (Android apps) can be done from any of the platfoms but exporting IPA (iOS apps) can only be achieved with a Mac device. Here's a breakdown of the platforms and the export formats supported by each:

Platform	Local Executable	APK (Android format)	IPA (iOS format)
Windows	✓	✓	✗
Mac	✓	✓	✓
Linux	✓	✓	✗

PUBLISHING YOUR APPS

You will need to sign up to an app store if you want to publish your apps. These are the three most popular mobile app stores:

- **Google Play Store**
- **Apple App Store**
- **Amazon App Store**

TIER 1 (BASIC) PUBLISHING FOR IOS

Exporting an application that will run on iPhone and iPad devices requires you to have a computer running Mac OS and an iOS developer account, which costs $99 per year. To enrol for an iOS developer account, visit the Apple Developer Site.

Distribution certificate
Once enrolled as an Apple Developer, the first step is to create a distribution certificate for your Apple account, here are the steps you need to follow to do this:

- Sign into your iOS developer account https://developer.apple.com/account/
- Select Certificates, Identifiers & Profiles.
- The menu on the left side displays Certificates, Keys, Identifiers, Devices and Provisioning Profiles.
- From the Certificates section select the Production link.
- Press the plus icon to create a new certificate.
- A list of options will be presented to you including Development and Production.
- At the bottom of this page is a link to an Intermediate Certificate.
- Download and install this certificate - Worldwide Developer Relations Certificate Authority.
- While on the same page from the Production listing select App Store and Ad Hoc and then press the continue button.
- Full instructions will be provided on the website explaining the next steps.
- Follow through this process and finally download and install your distribution certificate.

Once completed, your production certificate will be available for a period of 12 months and is used for all of the apps that you create. When this certificate expires you will need to renew it by returning to the page and following the prompts.

App requirements

Every application you want to distribute will require:

- An application ID
- An Ad Hoc or App Store distribution profile

Application ID

This is a unique identifier for your application and is also used to determine which services your application might need access to, for example iCloud, Push Notifications, etc. To create an application ID:

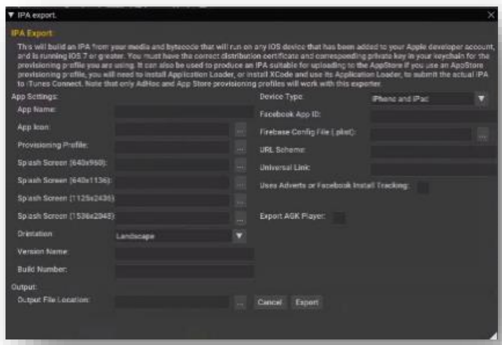
- Sign into your developer account.
- Select Certificates, Identifiers & Profiles.
- From the menu on the left look for the Identifiers listing and select App IDs.
- Press the + button to create a new App ID.
- You will be presented with a page asking you to fill in the App ID description, App ID Prefix, App ID Suffix and App Services.
- Enter a name for your app in the App ID Description section.
- Typically, the App ID Prefix will only have one option available - your latest distribution certificate.
- For the App ID Suffix select Explicit App ID and create a Bundle ID using a reverse domain name style string e.g. *uk.co.mywebsite.mygame*
- From the App Services listing tick any relevant functionality that your app may link to.
- Finally press the Continue button to create your Application ID.

Ad Hoc or App Store distribution profile

The next stage is to create a distribution certificate that can be used to install your app on a device:

- Sign into your developer account.
- Select Certificates, Identifiers & Profiles.
- From the menu on the left look for Provisioning Profiles and select Distribution.
- Press the plus button to create a new profile.
- A page will be displayed showing a number of options for Development and Distribution.
- From the Distribution section you can either select App Store or Ad Hoc.
- The App Store selection allows your app to be uploaded to the App Store where you can use programs like TestFlight to easily distribute and test your app before release.
- The Ad Hoc option lets you install your app on a device manually. If you choose this option you will need to add any devices you want to support into a list that can be found in the Devices, All section. Visit this page and follow the instructions to add devices you want to support.
- Select the deployment option and press the Continue button.
- At this point you will be prompted to select the App ID that you want to use.
- Pressing Continue will display another page where you need to select the distribution certificate being used, typically only one option will be present. Select this and press Continue.
- If you selected an Ad Hoc profile you will be asked to choose which devices the app can run on.
- Now you can download the provisioning profile. Your AppGameKit Studio project will need to reference this file, so place it in a convenient location.

Exporting an app


IPA export.

IPA Export

This will build an IPA from your media and bytecode that will run on any iOS device that has been added to your Apple developer account, and is running iOS 7 or greater. You must have the correct distribution certificate and corresponding private key in your keychain for the provisioning profile you are using. It can also be used to produce an IPA suitable for uploading to the AppStore if you use an AppStore provisioning profile, you will need to install Application Loader, or install XCode and use its Application Loader, to submit the actual IPA to iTunes Connect. Note that only AdHoc and App Store provisioning profiles will work with this exporter.

App Settings:
App Name:
App Icon:
Provisioning Profile:
Splash Screen (640x960):
Splash Screen (640x1136):
Splash Screen (1125x2436):
Splash Screen (1536x2048):
Orientation: Landscape
Version Name:
Build Number:
Output:
Output File Location:

Device Type: iPhone and iPad
Facebook App ID:
Firebase Config File (.plist):
URL Scheme:
Universal Link:
Uses Adverts or Facebook Install Tracking:
Export AGK Player:

With the Apple Developer account all set up and the App Id and Distribution profile in place, you're now ready to export your app. Open AppGameKit Studio (Mac version only) and then open your project. From the File menu select the Export Project to iOS. A dialog box will be displayed where you will need to fill in the majority of fields aside from those listed in the optional section.

The executable file type on iOS is called an IPA file (iOS App Store Package). Due to Apple rules you can only create IPAs by using a Mac desktop or laptop. As usual you can access the Export option from the File menu where you will find both APK and IPA export options.

The Export system in AppGameKit Studio will build an IPA from all your media and bytecode. This can then be run on any iOS devices that have been added to your Apple developer account and is running iOS7 or greater. You must have the correct distribution certificate and corresponding private key in your keychain for the provisioning profile you are using. It can also be used to produce an IPA suitable for uploading to the AppStore if you use an Appstore provisioning profile you will need to install Application Loader, or install XCode and use its Application Loader, to submit the actual IPA to iTunes Connect. Note that only AdHoc and App Store provisioning profiles will work with this exporter.

App Settings

App Name	The name of the app. You must only use A-Z,0-9, spaces and underscore characters in the name.
App Icon	Must be at least 1024x1024 pixels, PNG only, and must not contain transparency.
Provisioning Profile	Provisioning profile downloaded from your Apple developer account containing the list of devices this build will work on (or the App Store).
Splash Screen (640x960)	Displayed whilst your app is launching on iPhone 4 and below, should be at least 640x960. Landscape splash screens should be rotated 90 degrees clockwise so the bottom of the image is to the left.
Splash Screen (640x1136):	Displayed during app launching on iPhone 5 and above, should be at least 640x1136. Landscape splash screens should be rotated 90 degrees clockwise so the bottom of the image is to the left.
Splash Screen (1125x2436)	Displayed whilst your app is launching on iPhoneX, should be at least 1125x2436. Landscape splash screens should be rotated 90 degrees clockwise so the bottom of the image is to the left.
Splash Screen (1536x2048)	Displayed whilst your app is launching on iPad, should be at least 1536x2048. Landscape splash screens should be rotated 90 degrees clockwise so the bottom of the image is to the left.
Orientation	Choose the initial orientation of your app, landscape, portrait or both. You can alter it later in the code using the command *SetOrientationAllowed*
Version Name	The version name of this build. Allowed characters are 0-9 and . only, this must match the version number you use in iTunes Connect
Build Number	The build number of this exported build. Allowed characters are 0-9 and . only.

Output

Output File Location	The location to save the final IPA file.

Device Type	"iPhone and iPad", "iPhone only", "iPad only" - Choose what type of devices will be able to see this app on the AppStore.
Facebook App ID:	(Optional) If you use the Facebook commands you must put the Facebook App ID here.
Firebase Config File (.plist)	(Optional) The GoogleService-Info.plist file you downloaded when setting up your Firebase project, this is different for every app.
URL Scheme	The URL scheme that can be used to open your app from a browser link, e.g. a URL scheme of \"myapp\" would allow links to myapp://sometext to open your app with \"sometext\" being available to your app through GetURLSchemeText(). On iOS this must be unique to your app.
Universal Link	A Universal Link can be used to open your app from a URL link, e.g. \"https://www.appgamekit.com/app/\" will open the AppGameKit Player. In this field you only need to enter the domain, e.g. \"www.appgamekit.com\" and Apple will fetch the valid links from the \"https://www.appgamekit.com/apple-app-site-association\" file.
Uses Adverts or Facebook Install Tracking	Tick this box if your app uses the AdMob, Amazon Ads, or Chartboost commands, or the Facebook install tracking command. This will include the Advertising Identifier (IDFA) in your exported app. If you only use the Facebook commands but NOT the Facebook install tracking then you should not tick this box.
Export AGK Player:	Check this if you want to generate the AppGameKit Player App.

When you have filled in the relevant sections press the Export option. When the process is complete an IPA file will have been generated.

Installing an app using an Ad Hoc profile

When using an Ad Hoc profile, you will need to manually install the IPA by connecting your iOS device to your computer, opening iTunes, drag the IPA file into the library and then sync the device. Alternatively you can install the latest version of Xcode (Mac OS only) and from within it select the Window menu, then Devices and Simulators, where you will see any devices connected. From here you can drag the IPA file to the list of apps, which will install it to your device.

Installing an app using an App Store profile

An app listing will need to be generated prior to uploading any files.

- Login to App Store Connect and select My Apps. Press the plus button and select New App.
- Fill in the details and select the Bundle ID (App ID) from the listing that you created in the developer portal. More details on App Store Connect can be found HERE.
- The next step is to install Xcode (Mac OS only). Once this is installed search for Application Loader. Run this app and sign into your developer account.
- From within Application Loader press the choose button and navigate to and select your IPA. Follow the on-screen prompts to complete the upload.
- Once the upload is complete a short period of automated processing may be necessary.
- When this has finished you will be automatically emailed with a message confirming that app processing has ended. At this point return to App Store Connect and select My Apps and your app. Links will display for App Store, Features, TestFlight and Activity.
- Select the TestFlight option to see your uploaded build.
- Click on the build and enter any relevant information. Finally, from the TestFlight page select App Store Connect users and add people into the testing list, who will be emailed and invited into the testing process. Anyone testing your app will be invited to install TestFlight from the App Store. Once TestFlight is installed invited users can install your app.

More details on TestFlight can be obtained HERE

Notes

- The Apple website has comprehensive tutorials on every step of the development and submission process when submitting applications, and is highly recommended reading.
- YouTube is a great source of video tutorials on the various steps involved in registering an Apple account, installing your certificates and going through the validation process.

TIER 1 (BASIC) PUBLISHING FOR ANDROID

Exporting an application that will run on Android devices and for publishing to the Google Play and Amazon stores requires you to have a computer running Windows, Mac or Linux.

A Google Play developer account for a one-off fee of $25 is required for distribution on the Google Play Store. To enroll for a Google Play developer account, visit HERE. Developers can register an Amazon account for free HERE.

Local distribution

This method of distribution allows you to deploy apps yourself either through a website or email, without needing any signing certificates. This is a convenient option for testing your app, however, anyone wanting to install your app will need to go to their device security settings and select the option that allows apps from unknown or untrusted sources to be installed. To export an app using local distribution go directly to the section on exporting an app below.

Store distribution

Apps that you want to distribute on stores such as Google Play and Amazon will require your exported app to be signed with a security certificate, that can be generated directly from within AppGameKit Studio.

To generate this certificate open AppGameKit Studio, go to the Tools menu and select the option to Generate Keystore File. You will be presented with a dialog containing an explanation of the keystore file and several fields, (some of which are optional) that need to be filled in. Only the password and output file location fields are mandatory.

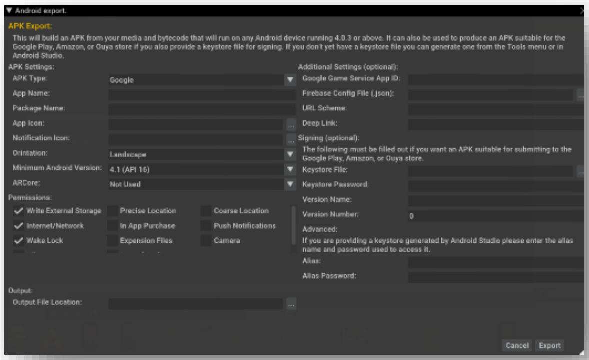
- Full Name - your name
- Company Name - your company name (if you have one)
- City - city where you are based
- Two Letter Country Code - country code e.g. UK
- Create a Password - enter a password
- Re-enter Password - confirm your password
- Output file location - the output location for the keystore file

When the form has been filled in select the Generate option to create your keystore file. The recommended approach is to create one keystore file that is used to sign all of your exported apps. Bear in mind this keystore is an essential part of the export process and any apps submitted with this keystore can only receive updates when signed with the same file.

Exporting an app

The executable file type for Android is called an APK (**A**ndroid **P**ac**K**age). When it's time for you to export a projects to this format you will need to fill out a number of fields. If you plan to publish your app to the Google or Amazon app stores then you need to register with those stores first (see above).

Here are all the Android Export settings. The left side listed first:



This will build an APK from your media and bytecode that will run on any Android device running 4.0.3 or above. It can also be used to produce an APK suitable for the Google Play, Amazon, or Ouya store if you also provide a keystore file for signing. If you don't yet have a keystore file you can generate one from the Tools menu or in Android Studio.

APK Settings:
APK Type: Google
App Name:
Package Name:
App Icon:
Notification Icon:
Orintation: Landscape
Minimum Android Version: 4.1 (API 16)
ARCore: Not Used

Permissions:
✓ Write External Storage
✓ Internet/Network
✓ Wake Lock
Precise Location
In App Purchase
Expension Files
Coarse Location
Push Notifications
Camera

Output:
Output File Location:

Additional Settings (optional):
Google Game Service App ID:
Firebase Config File (.json):
URL Scheme:
Deep Link:

Signing (optional):
The following must be filled out if you want an APK suitable for submitting to the Google Play, Amazon, or Ouya store.
Keystore File:
Keystore Password:
Version Name:
Version Number: 0
Advanced:
If you are providing a keystore generated by Android Studio please enter the alias name and password used to access it.
Alias:
Alias Password:

APK Settings

APK Type	Sets the type of APK to export. You can choose Google, Amazon or Ouya. While Ouya is now an old format it does allow you to export a build that does not use Facebook or Advert features.
App Name	The name of the app. You must only use A-Z,0-9, spaces and underscore characters in the name.
Package Name	The package name is used by the stores. Valid characters to use are A-Z, full stop ".", underscore "_". The format should be: com.mycompany.my_app
App Icon	Must be at least 192x192, PNG format only and it can contain transparency.
Notification Icon	If your app makes use of notifications you can set the icon that appears in the device's notification bar. Must be at least 96x94, PNG format only. It must be all white with the shape defined by the alpha channel.
Orientation	Choose the initial orientation of your app, landscape, portrait or both. You can alter it later in the code using the command *SetOrientationAllowed*
Minimum Android Version	This only applies to Google and Amazon exporting. For maximum compatibility choose 4.0.3, or restrict your app to a chosen minimum version. If you choose 6.0 (API 23) or above then you must use the *RequestPermission* command to request any required permissions before using certain commands.
ARCore	Sets whether your app uses or requires ARCore to be installed on the user's device. Requiring ARCore will limit your app to Android 7.0 or above.

Permissions

Write External Storage	This permission is used for camera capture images, shared variables and "raw:" file access.
Internet/Network	Needed for network or Internet access.
Wake Lock	Required to prevent the device from sleeping.
Vibration	Used to allow vibration of the device.
Precise Location	Needed by the GPS commands for exact location.
In App Purchase	The In-App purchase commands require this permission.
Expansion Files	Adds the Get Accounts and Google Licensing permissions used by the expansion file commands.
Record Audio	Used to allow recording from the microphone when recording the screen.

Coarse Locations	The GPS commands need this for rough location.
Push Notifications	Push Notification commands require this.
Camera	Used by the *SetDeviceCameraToImage* command.

Output

Output File Location	The location of your device where the APK will be save to.

Additional Settings (these are optional)

Google Game Service App ID	The 12-digit application ID assigned by Google Game Services to your app.
Firebase Config File (.json)	The google-services.json file that you downloaded when setting up your Firebase project, this is different for every app.
URL Scheme	The URL scheme that can be used to open your app from a browser link, e.g. a URL scheme of "myapp" would allow links to myapp://sometext to open your app with "sometext" being available to your app through GetURLSchemeText().
Deep Link	The deep link that can be used to open your app from a browser link, e.g. a deep link of "http://www.appgamekit.com/app" would allow all links beginning with those characters to open your app with the entire being available to your app through GetURLSchemeText().

Signing (optional)

The following must be filled out if you want an APK suitable for submitting to the Google Play, Amazon or Ouya store.

Keystore File	If you don't have this you can generate one from the Tools drop-down menu.
Keystore Password	The password you chose when you created the Keystore file.
Version Name	e.g. 1.0.1.
Version Number	Must be an integer and must be higher than any value already submitted to the store.

Advanced

If you are providing a keystore generated by Android Studio please enter the alias name and password used to access it.

Alias	The name of the alias used when the keystore file was generated.
Alias Password	The password for the Alias.

When you are ready to export an app that can be deployed to Android devices. Open AppGameKit Studio and then your project. From the File menu select the *Export Project to Android* option.

When you have filled in the relevant sections press the Export option. When the process is complete an APK file will have been generated.

Installing an app manually without signing
As mentioned earlier if your app is not signed anyone wanting to install it must alter the security settings on their device to allow unsigned or unknown apps to be installed. Once this is done your app can be installed through a variety of methods:

- Connect the test device to your computer using a USB cable and copy the exported APK across to it. From the device use a file explorer (either built in or one from the stores), navigate to the location of the APK, select and install it.
- Upload the APK from your computer to Google Drive and either share a link or access it directly from Google Drive on your device and install.

Installing an app with signing
Prior to installing a signed app a store listing will need to be generated on https://play.google.com/apps/publish for Google or https://developer.amazon.com/ for Amazon.

For Google click on the create application button and follow the instructions. At least for the testing phase it's not necessary to fill in every single detail. You can just deal with the basic information along with pricing & distribution and the content rating.

With the basic information filled out you can now proceed to uploading your APK. To do this go to Release management and then App releases, where you will be presented with a screen offering you options for production track, beta, alpha and internal test track. For testing purposes it's a good option to select the internal test track, click on manage then select create release. From here you are asked to select your APK to be uploaded. You can also create a testing list of up to 100 testers who can install your app using the provided link. When your app is ready for release you can return to this section and select the release to alpha option and promote it to the next stage of release, from there you can switch it to the production track and make it live on the store.

For Amazon once signed in go to the developer console, click on Apps & Services, select My Apps and then the Add New App button, then follow the prompts to fill in the required information.

With the basic information filled out you can now proceed to uploading your APK. To do this select your app listing and then click on Live App Testing. Click on the New Test option and follow the prompts. Once your app has completed the processing stage it will be available to any testers you list in the Live App Testing section via email invite. When your testing has completed you can promote the Live App Testing version to upcoming and at this point can submit to Amazon to be deployed onto the app store.

TIER 2 – THE C++ LIBRARIES

While the main focus of AppGameKit Studio is the BASIC Script language, we also provide C++ libraries that give you access to all the commands that can then be intergrated into your prefered development environemnt.

GETTING STARTED WITH TIER 2

Tier 2 provides you with the ability to use the AppGameKit Studio command set within a C++ application. This is an ideal option in many cases, for example:

- For situations where you want to get the best possible speed on a device. Games written in C++ will run faster than those using Tier 1, as the games will be running natively, unlike Tier 1, which will run games using an interpreter.
- In cases where you have an existing codebase that is already in C++ and you want to reuse this with AppGameKit Studio.
- If you're already familiar with C++ then you may prefer to continue developing a game in this language instead of having to learn the Tier 1 language.

The downside is that compared to Tier 1 there is an extra level of difficulty in terms of setting up projects and compiler settings and so on. However, if you are already familiar with C++ then it's highly likely that you'll be up and running in Tier 2 within a short space of time.

A collection of template projects and C++ libraries are provided for Tier 2 applications. You can choose where to install these by choosing the menu option *Tools/Install Additional Files*. You will be prompted for a location to install these files. Once installed you will see the following folders inside the folder where you choose to install:

Apps	In this folder are the templates for creating native projects on various platforms, plus the interpreter project for recompiling the AppGameKit Studio Player.
Bullet	The library files for the Bullet 3D Physics engine.
Common	This holds the include files you will need to point to when compiling, the Apps projects already reference this folder with a relative path.
FirebaseSDK	The library files for the Firebase SDK.
Platform	Contains the native libraries for various platforms that you will need to link to when compiling, the Apps projects already reference this folder with a relative path.
Showcase	Example games that come for free with AppGameKit Studio, in binary form only.
Tutorials	Several tutorials to get started.

Templates
The apps folder contains projects suitable for the current platform, for example the Windows version contains projects for Windows and Android, whilst the Mac version contains projects for Mac, iOS, and Android.

All of these projects use relative paths to link to libraries and include files, so if you create a copy of a project and move it elsewhere please bear in mind you will need to alter these paths.

Library files
Most platforms contain a single library compiled in release mode that can be compiled into both Debug and Release projects. The exception is Windows which has both a Debug and Release library which must match your project's current build configuration. For example, if you enter the Windows folder, you will see a folder named Lib, go into here and there will be two folders; Debug and Release that contain the library files. Your debug configuration should link to the one in the Debug folder and your release configuration should link to the one in the Release folder.

WINDOWS

This guide explains the process involved in creating projects when using Tier 2 on the Microsoft Windows platform. AppGameKit Studio currently supports Microsoft Visual Studio 2015 and Microsoft Visual Studio 2017, which is available to download from https://visualstudio.microsoft.com/.

Libraries and include files

The template projects and any of your own custom projects are dependent on several library and include files. Their install locations are:

Include Files
```
\common\include\
\common\Collision\
\bullet
\bullet\BulletCollision\CollisionShapes
```

Libraries (VS 2015)
```
\platform\windows\Lib\VS2015\Debug\
\platform\windows\Lib\VS2015\Release\
```

Libraries (VS 2017)
```
\platform\windows\Lib\VS2017\Debug\
\platform\windows\Lib\VS2017\Release\
```

Templates

Several templates are included that provide a starting point for your applications. These can be found within -

```
\apps
```

The key templates are -

- template_windows_vs2015 - 32 bit template for VS 2015
- template_windows_vs2015_64 - 64 bit template for VS 2015
- template_windows_vs2017 - 32 bit template for VS 2017
- template_windows_vs2017_64 - 64 bit template for VS 2017

All of these projects are set up with relative links to the AppGameKit Studio libraries and include files and are ready to compile. Making new projects outside this location or moving the existing projects to another location will result in you needing to update library and include links within the project settings to ensure the app compiles.

- Make a copy of the appropriate project and place it within the apps folder, then open the project within Visual Studio.
- Attempt to compile the project by going to the Build menu and selecting Build Solution. If all goes well a message will appear in the Output window showing Build succeeded. Any failure at this point may be due to an issue with the platform toolset, that can be fixed by entering the project settings, going to the general section and selecting an available platform toolset.

The project will consist of three main files - core.cpp, template.cpp and template.h. Core.cpp is the starting point for your application as a whole, which in most cases can be ignored and is only relevant for more advanced users. Template.h contains a class declaration for your app and template.cpp contains the definition, with three functions already defined - Begin, Loop and End. Begin is the entry point for your application and is used for initial setup. Loop will be called every cycle and should be the location of your logic etc. Finally, End is called when the app closes down and can be used to free up resources.

MAC OS X

This guide explains the process involved in creating projects when using Tier 2 when using the Macintosh OS X platform. In order to compile projects on the Mac you will need Xcode 10 or higher running Mac OS X 10.13.6 or later. Xcode is available for free by visiting https://developer.apple.com/xcode/ Alternatively you can obtain it from within the Mac App Store by searching for Xcode.

All of the relevant files for Tier 2 development are bundled within the main package of AppGameKit Studio. To access them navigate to the install location for AppGameKit Studio e.g. Applications/AGK and right click on the icon for AppGameKit Studio and select Show Package Contents. From here enter the Contents folder and then the Resources folder where you will see a Tier 2 folder with libraries, include files and examples projects for Mac OS X development.

Libraries and include files
The template projects and any of your own custom projects are dependent on several library and include files. Their install locations are:

Include Files
```
/common/include/
/common/Collision/
/bullet
/bullet/BulletCollision/CollisionShapes
```

Libraries
```
/AppGameKitStudio.app/Contents/Resources/platform/mac/Lib/Release
```

Templates
A template is included that provides a starting point for your application. This can be found within

```
/AppGameKitStudio.app/Contents/Resources/Tier 2/apps
```

The main project of interest is called template_mac. This project is set up with relative links to the AppGameKit Studio libraries and include files and are ready to compile. Making new projects outside this location or moving the existing projects to another location will result in you needing to update the library and include links within the project settings to ensure the app compiles.

- Make a copy of the template_mac and place it within the apps folder
- Now open the project within Xcode.
- Attempt to compile the project by going to the Project menu and selecting Build.
- When your project has built a message will show up saying it has succeeded. From here you can run the app by going to the Product menu and selecting the Run option.

The project will consist of three main files - core.mm, template.cpp and template.h. Core.mm is the starting point for your application as a whole, which in most cases can be ignored and is only relevant for more advanced users. Template.h contains a class declaration for your app and template.cpp contains the definition, with three functions already defined - Begin, Loop and End. Begin is the entry point for your application and is used for initial setup. Loop will be called every cycle and should be the location of your logic etc. Finally, End is called when the app closes down and can be used to free up resources etc.

ANDROID

This guide explains the process involved in creating projects when using Tier 2 for the Android platform. As the requirement is for Android 2.3.3 and above, you are able to code in native C/C++, rather than Java. Please note this guide is intended to be used when compiling on Windows.

You will need to download the following files and packages. This assumes you have a 64-bit computer, if you have a 32-bit computer then download the 32-bit (x86) versions instead.

Java JDK
Go to this web page http://www.oracle.com/technetwork/java/javase/downloads/index.html and download the latest version of the JDK. Click the download button next to the JDK, accept the license, and download the Windows x64 package. Once the download is complete proceed to install the software.

Android NDK
Go to https://developer.android.com/ndk/downloads/index.html and download the 64-bit package for Windows. Install these files to a location that does not contain spaces e.g. d:\AGK\NDK is acceptable whereas d:\my software\AGK\NDK is not.

Android Studio
Go to http://developer.android.com/sdk/index.html and download the latest version of Android Studio for your platform. Once downloaded run the installer and follow the on-screen prompts.

Note: The AppGameKit Studio Tier 2 files must be installed in a path with no spaces, otherwise the NDK will fail to compile the necessary files.

Running Android Studio
Launch Android Studio and follow these instructions:

- Android Studio should automatically detect your Java installation, if not you will get an error message saying that Android Studio can't find it and ask you to set a JAVA_HOME variable. Double check your Java installation and set up a JAVA_HOME variable if necessary.
- When Android Studio first runs it will present an SDK Components Setup dialog, click *Next* or *Finish* and let it install the default components.
- When completed you should see a list of options such as *Start a new Android Studio Project* and *Open an existing Android Studio Project*, select *Configure* and then *SDK Manager*.
- In the SDK Platforms tab Android Studio will have automatically installed the latest SDK Platform, but you need to install a specific version. If the latest version is not 8.0 (API 26) then tick the box next to Android 8.0 (API 26). You only need the Platform option, not the Google APIs or any of the others.
- Now switch to the SDK Tools tab and tick *Android Support Repository*, *Google Play Services*, and *Google USB driver*, then click OK and let it install.
- Come back out of the Configure option and choose *Open an existing Android Studio Project*, browse to your AGKStudio Tier 2 folder, then apps, select the *template_android_google* folder and click ok.
- When the project first loads, Gradle will attempt to sync and build the project, Grade is the build manager that ships with Android Studio. You will likely get some messages from Grade Sync saying it failed to find the correct Build Tools version, click the provided link to install the missing version.
- Now Android Studio should be working without error, however it will not produce a valid application until we compile the NDK libraries for our project.
- Minimize Android Studio and browse to the location you installed the AGKStudio Tier 2 files, then go to apps\template_android_google\AGKStudioTemplate\src\main
- In here you will see a file named jniCompile.bat, right click on it and choose Edit

- Change the line that starts set NDKBUILDCMD= to point to your NDK location, for example set NDKBUILDCMD="C:\AndroidDev\android-ndk-r16b\ndk-build", save this file and close it, then double click it to run the NDK compile process for this project. The NDK folder name may vary based on which version of the NDK you downloaded. The folder name itself is not important as long as it points to the NDK you downloaded.
- It will compile three libraries, one for arm64-v8a, one for armeabi-v7a, and one for x86, these cover the three most popular architectures that run Android. The libraries will be placed in the appropriate folders for Android Studio to pick up next time it builds an APK of your project.
- When it is done it should end with a line starting [x86] Install, if not then there was an error and you should check the log.txt file in the same folder as the jniCompile.bat file for details
- Now return to Android Studio and choose Build->Make Project it should display the Grade Console window with its progress and end with a BUILD SUCCESSFUL message. You can attempt to run this project on your device if you have already enabled it for USB Debugging, if not continue to the next section, or you can create a virtual device to test the project in an emulator.
- The template_android_google app should display a pale blue screen with the framerate displayed near the top left. It is not recommended that you make changes to this project as it can serve as the template for future projects. The recommended approach is to copy the template project and make changes to the copy, see the Creating a New Game Project for details.

Running on a device
The following steps will show you how to setup your device for testing your apps.
Connect your Android device (your phone or tablet) to your PC by USB and turn on USB Debugging in Android's settings. To turn on USB Debugging:

- Plug in your device.
- Open Settings, scroll down to and tap on Developer Options, and tap the on-off switch at the top right.
- Tap USB Debugging and accept the warning to enable it.
- This should trigger windows to re-detect your device with additional device components, this is where the Google USB Driver comes in that we downloaded earlier. Windows may automatically detect the debug portion of your device successfully, or it may require you install the Google USB Driver to detect it correctly.

More detailed instructions on USB Debugging and how to install the USB debug driver vary by device, and are outside the scope of this guide, but there are plenty of tutorials online.

You will also need to enable the Unknown Sources option in the security section so that apps can be installed that have not been downloaded through Google Play.

Once your device is setup to receive debug apps you can click the Run icon in Android Studio and it should automatically detect your connected device and upload your newly compiled app to it.

Creating a new project

The following steps will show you how to set up a new project. Once you've completed all the preparation sections of this guide, this is where you should start from each time you want to make a new project.

- Navigate to your AppGameKit Studio Tier 2 files folder, open the apps folder and make a copy of the template_android_google folder. This will be your new project folder, and you can rename it as you wish as long as you don't use spaces.
- Note that for simplicity your new folder should remain within the apps folder as it contains relative paths to other areas of the AppGameKit Studio Tier 2 folder, notably the platform/android/jni folder. Moving it to another location will require project settings to be updated.
- When you open Android Studio it will default to opening the last project you had open, go to File->Close Project to return to the main menu if you no longer want to work with it, you can then open your new project with Open an existing Android Studio Project and browsing to the new project folder you created.
- NDK support in Android Studio is still experimental so we use the jniCompile.bat file mentioned earlier, but you can still edit the C++ files (those with the .cpp or .h extensions) inside Android Studio. Be sure to call the jniCompile.bat file after you make any changes to those files.
- To change the package name for your app open the AndroidManifest.xml file and edit the package field from its default of com.mycompany.mytemplate. Note that when you do this you will need to go through the other files and change any instance of com.mycompany.mytemplate to your new package name, you will get an error if you forget to do this.
- Leave any instances of com.thegamecreators.agk_player as they are, it does not affect the final app and is used to tie together lots of parts of the app, so handle with care.
- Open the build.gradle file for the AGKStudioTemplate module and edit the applicationId field from its default of com.mycompany.mytemplate to your chosen package name.
- To change the name of the app, open the res/values/strings.xml file and edit the app_name field.
- To write your app code you only need to modify the template.h and template.cpp files, these files are only read by the Android NDK when you run jniCompile.bat (or compile manually using the command line)
- If you wish to add more .cpp files you will need to edit the following file using notepad: AGKStudioTemplate\src\main\jni\Android.mk Look for the LOCAL_SRC_FILES := line and add additional .cpp files there, save it, and run jniCompile.bat again to recompile.
- To add media to your project, create an assets folder at [project_folder_name]\AGKStudioTemplate\src\main\assets replacing [project_folder_name] with your renamed folder. Note that in Tier 2 a folder named media is not required and you can place images, sounds, etc, directly in the assets folder. If you do use a media folder then remember to use agk::SetFolder("/media") near the start of your program to make it behave more like Tier 1.

IOS

This guide explains the process involved in creating projects when using Tier 2 when using the iOS platform. In order to compile projects on the Mac you will need Xcode 10 or higher running Mac OS X 10.13.6 or later. Xcode is available for free by visiting https://developer.apple.com/xcode/ Alternatively you can obtain it from within the Mac App Store by searching for Xcode.

All of the relevant files for Tier 2 development are bundled within the main package of AppGameKit Studio. To access them navigate to the install location for AppGameKit Studio e.g. Applications/AGKStudio and right click on the icon for AppGameKit Studio and select Show Package Contents. From here enter the Contents folder and then the Resources folder where you will see a Tier 2 folder with libraries, include files and examples projects for iOS development.

Libraries and include files
The template projects and any of your own custom projects are dependent on several library and include files. Their install locations are -

Include Files
```
/common/include/
/common/Collision/
/bullet
/bullet/BulletCollision/CollisionShapes
```

Libraries
```
/Resources/platform/apple/Lib/Release
```

Templates

A template is included that provides a starting point for your application. This can be found within:

```
/Resources/Tier 2/apps
```

The main project of interest is called template_ios. This project is set up with relative links to the AppGameKit Studio libraries and include files and are ready to compile. Making new projects outside this location or moving the existing projects to another location will result in you needing to update library and include links within the project settings to ensure the app compiles.

Make a copy of the template_ios and place it within the apps folder, then open the project within Xcode. Attempt to compile the project by going to the Project menu and selecting Build. When your project has built a message will show up saying it has succeeded. From here you can run the app by going to the Product menu and selecting the Run option. You can choose to test on the simulator or an actual device by selecting the target listing in Xcode.

The project will consist of three main files - core.mm, template.cpp and template.h. Core.mm is the starting point for your application as a whole, which in most cases can be ignored and is only relevant for more advanced users. Template.h contains a class declaration for your app and template.cpp contains the definition, with three functions already defined - Begin, Loop and End. Begin is the entry point for your application and is used for initial setup. Loop will be called every cycle and should be the location of your logic etc. Finally End is called when the app closes down and can be used to free up resources etc.

FREQUENTLY ASKED QUESTIONS

Q: CAN I RUN MY APPGAMEKIT CLASSIC PROJECTS IN APPGAMEKIT STUDIO?

A: Yes, absolutely! The command set is the same, so your Classic projects will run just fine in Studio and they will be able to run with the new Vulkan engine too.

Q: HELP, MY DEBUG WINDOW LAYOUT HAS BECOME CORRUPTED!

A: If this happens you can reset the system by viewing these folders and deleting the file *window_position.dat* if you see it in:

```
c:\Users\USERNAME\AppData\Local\AGKApps\Windows\

c:\Users\USERNAME\AppData\Local\AGKApps\Windows64\
```

When you have deleted the file, re-run AppGameKit Studio and all should be back to default settings.

Q: WHEN WILL MAC & LINUX USERS GET THE VULKAN ENGINE?

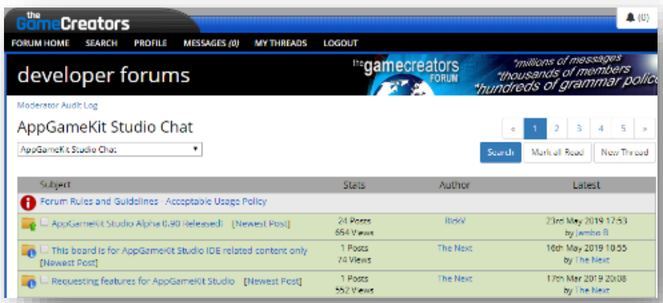
A: We're aiming to have these released a few weeks after launch. It just depends how easy or hard the work will be. We'll keep the community updated in the online forums and with news announcements on the main website and Steam news section.

Q: WHERE TO GET FURTHER HELP?

There are lots of online materials to help you study and learn more about programming and game creation:

TheGameCreators Forums

The forums are a great community resource of advice, help, demos, source code and more. You're free to visit the forums and read them at your leisure and you'll get even more value from them if you register an account with TheGameCreators and post your own questions to ask for help or to share your views and code. We look forward to hearing from you!



AppGameKit Studio Chat

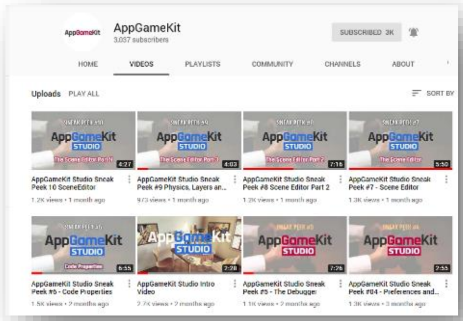
AppGameKit Studio Chat ▾

Subject	Stats	Author	Latest
Forum Rules and Guidelines - Acceptable Usage Policy			
AppGameKit Studio Alpha 0.90 Released! [Newest Post]	24 Posts 654 Views	RickV	23rd May 2019 17:53 by Jambo B
This board is for AppGameKit Studio IDE related content only [Newest Post]	1 Posts 74 Views	The Next	16th May 2019 10:55 by The Next
Requesting features for AppGameKit Studio [Newest Post]	1 Posts 552 Views	The Next	17th Mar 2019 20:08 by The Next

Steam Community Forums

We also have a community forum on the Steam platform. It's an ideal place for Steam users to share their discussions about AppGameKit Studio. Join the Steam discussions HERE.

AppGameKit Official YouTube Channel



We've published many video tutorials on our YouTube channel and continue to do so. You will find that the earlier ones were made using AppGameKit Classic – there are still valid and fine to use in AppGameKit Studio. It's best to subscribe and click the bell if you want to be notified when new videos are released. There are getting started tutorials that show you how to load images and make sprites and then they cover making a simple shooter game and a full version of the classic Tetris puzzle game.

AppGameKit Facebook Group & AppGameKit Twitter Account

If you prefer to get your news and advice from social media, we have you covered with dedicated channels on Facebook and Twitter. Both channels post news and information on a regular basis so you can keep up to date with all things AppGameKit Studio!

AppGameKit Discord Group

The community runs a Discord Group and the main dev team post and reply on a regular basis.