How to create HTML5 games on the Raspberry Pi 4 using AGKTier2 (C++)

Download and build LLVM

Download from: https://github.com/llvm/llvm-project

I followed the instructions on how to build LLVM from :

https://emscripten.org/docs/building_from_source/index.html#building-llvm

This may take a little time (a few hours) to build on your raspberry pi 4.

Download and build Binaryen

I followed the instructions on how to build Binaryen from:

https://github.com/WebAssembly/binaryen#building

Download emscripten

Download from: https://emscripten.org/

Download nodejs

sudo apt-get install nodejs

You will need to install acorn using:

```
npm install acorn
```

At this stage you should have LLVM, Binaryen, emscripten, nodejs and the acorn module installed.

Setup emscripten config file

Run emscripten for the first time by typing ./emcc in to your terminal, this should create a hidden config file called .emscripten

```
sudo nano .emscripten
```

I use the following settings:

```
# This is used by external projects in order to find emscripten.
It is not used
# by emscripten itself.
EMSCRIPTEN_ROOT = '/home/pi/Dev/emscripten' # directory

LLVM_ROOT = '/usr/local/bin' # directory
BINARYEN_ROOT = '/home/pi/Dev/binaryen' # directory

# Location of the node binary to use for running the JS parts of
the compiler.
# This engine must exist, or nothing can be compiled.
NODE_JS = '/usr/bin/nodejs' # executable
```

Run emscripten again (./emcc) to detect your settings

Build libAGKHTML5.a

I assume you have already built the libAGKLinux.a library file, if not, you need to build it before you can build libAGKHTML5.a

From your AGKTier2 folder just type:

```
make
```

This should build libAGKLinux.a

Open up a terminal window from your AGKTier2 folder and type:

Use nano or your editor of choice.

```
sudo nano Makefile_html5
```

Edit the CC entry to point to emscripten.

My entry looks like this:

```
CC = /home/pi/Dev/emscripten/emcc
```

Before you build libAGKHTML5.a you need to add an entry to INC so the compiler can find the emscripten include files.

Change the entry to where you have saved emscripten

```
-I/home/pi/Dev/emscripten/cache/sysroot/include/emscripten
```

Save the file.

Before we can create libAGKHTML5.a we need to modify HTML5Core.cpp

The file is located in: **AGKTier2/platform/html5/Source**

Open the file in your editor of choice.

Locate the following functions:

void agk::PlatformInitGL(void* ptr)
void agk::PlatformInitConsole()
void cJoystick::DetectJoysticks()
void cJoystick::PlatformUpdate()

In each function you should see:

```
int numJoysticks = emscripten_get_num_gamepads();
```

Add the following lines:

```
int gamepadData = emscripten_sample_gamepad_data();
if (gamepadData == EMSCRIPTEN_RESULT_NOT_SUPPORTED) return;
```

According to the emscripten site you need to call this function before calling either of the functions emscripten_get_num_gamepads() or emscripten_get_gamepad_status().

For example:

```
void cJoystick::PlatformUpdate()
{
     int gamepadData = emscripten_sample_gamepad_data();
     if (gamepadData == EMSCRIPTEN_RESULT_NOT_SUPPORTED) return;

     int numJoysticks = emscripten_get_num_gamepads();
     if ( numJoysticks == EMSCRIPTEN_RESULT_NOT_SUPPORTED ) return;
```

Search for all the functions and do the same for each function.

Next, Search for all entries of the deprecated Pointer_stringify and replace with UTF8ToString

Build libAGKHTML5.a from your AGKTier2 folder and type:

```
make -f Makefile_html5
```

Hopefully you should have libAGKHTML5.a located at:

```
AGKTier2/platform/html5/Lib/Release
```

Before we start creating a simple html5 program we need to setup an html5 template.

In AGKTier2 folder, type:

```
mkdir template_html5
```

Located in AGKTier2/apps/interpreter_html5 is a file called Core.cpp

Copy Core.cpp to template_html5

Use your editor of choice and open Core.cpp

You should see a few include directives at the top.

```
// includes
#include "agk.h"
#include "interpreter.h"

#define GLFW_INCLUDE_ES2
#include "GLFW/glfw3.h"
#include <SDL.h>
#include "emscripten.h"
#include "emscripten/html5.h"
```

Replace with:

```
// includes
#include "agk.h"

#define GLFW_INCLUDE_ES2
#include "GLFW/glfw3.h"
#include <SDL.h>
#include "emscripten.h"
#include "emscripten/html5.h"
#include "template.h"
```

We will now setup template.cpp and template.h

Create the file template.h and add the following lines of code.

```
#ifndef _H_AGKTEMPLATE_
#define _H_AGKTEMPLATE_

// Link to AGK files
#include "agk.h"

#define DEVICE_WIDTH 640
#define DEVICE_HEIGHT 480
#define DEVICE_POS_X 32
#define DEVICE_POS_Y 32
#define FULLSCREEN false

// used to make a more unique folder for the write path
#define COMPANY_NAME "My Company"

// Global values for the app
class app
{
     public:

          // constructor
          app() { memset ( this, 0, sizeof(app)); }

          // main app functions
          void Begin( void );
          int Loop( void );
          void End( void );

          // Needed for HTML5
          unsigned int g_dwDeviceWidth;
          unsigned int g_dwDeviceHeight;
          unsigned int g_dwFullScreen;
          char g_pWindowTitle [ 512 ];
};

extern app App;

#endif

// Allow us to use the LoadImage function name
#ifdef LoadImage
 #undef LoadImage
#endif
```

Create the file template.cpp and add the following lines of code:

```
#include "template.h"

app App;

void app::Begin(void)
{
    agk::SetVirtualResolution(640, 480);
    agk::SetSyncRate(30, 0);
    agk::SetScissor(0, 0, 0, 0);
    agk::UseNewDefaultFonts(1);
}

int app::Loop (void)
{

  agk::Print("Hello HTML5");

  agk::Sync();

  return 0; // return 1 to close app
}

void app::End (void)
{

}
```

Now we need to create a Makefile - (I altered the original Makefile)

```
CC = /home/pi/Dev/emscripten/emcc
ODIR = build/obj
INC = -I../../common/include -I../../common -I../../bullet
-I../../bullet/BulletCollision/CollisionShapes
CFLAGS = -O2 -std=c++11
ifeq ($(2D),1)
      CFLAGS += -DAGK_2D_ONLY
endif
LDFLAGS = -L../../platform/html5/Lib/Release

_OBJS = Core.o template.o

OBJS = $(patsubst %,$(ODIR)/%,$(_OBJS))

all: setup $(OBJS) AGKPlayer

setup:
      mkdir -p build/obj

$(ODIR)/%.o: %.cpp
      $(CC) -DIDE_HTML5 -c $(INC) -o $@ $< $(CFLAGS)

AGKPlayer:
      $(CC) $(OBJS) -o AGKPlayer.html $(LDFLAGS) -O2 --shell-file
../../platform/html5/Source/shell.html -s USE_GLFW=3 -s FULL_ES2=1 -s USE_SDL=1
-s WASM=1 -s ALLOW_MEMORY_GROWTH=1 -s DISABLE_EXCEPTION_CATCHING=0 -lAGKHTML5

clean:
      rm -rf build/obj/*
      rm -f AGKPlayer.html
```

I replaced the original agkshell.html with the emscripten shell.html – can be found in emscripten/src

We should now have a template_html5 folder with the following files:

```
Core.cpp
template.cpp
template.h
Makefile
```

In the template_html5 folder, type: `make`

If all went well the following files will be created:

```
AGKPlayer.html
AGKPlayer.js
AGKPlayer.wasm
```

You can test this out by using the built-in python http server to test your html5 example.

Depending on your version of python you can use:

python -m SimpleHTTPServer

or

python -m http.server

In your web browser type in http://localhost:8000

When creating more complex programs that use graphics and sounds you will need to add a media folder by adding the flag

```
--preload-file media
```

Add this flag after -lAGKHTML5

For example:

```
AGKPlayer:
	$(CC) $(OBJS) -o AGKPlayer.html $(LDFLAGS) -O2 --shell-file
../../platform/html5/Source/shell.html -s USE_GLFW=3 -s FULL_ES2=1 -s USE_SDL=1
-s WASM=1 -s ALLOW_MEMORY_GROWTH=1 -s DISABLE_EXCEPTION_CATCHING=0 -lAGKHTML5 --
preload-file media
```

this will add all files stored in media and add an additional file called AGKPlayer.data

Hopefully this tutorial is useful!

-TBoy