**Work in Progress Report**

**By: Deep Raithatha, Jash Pandya, Jose Rivas**

**GitHub URL: https://github.com/JoseRivas/FlightlessFury**

**Major developments/breakthroughs(reference specific code please):**

**Side Scrolling**

**Source:**

https://code.google.com/p/libgdx-users/wiki/ScrollingTexture

**Branch: master**

Last year we used 2 seperate backgrounds that loop through the screen.

This year we found that libgdx’s texture class has a setwrap feature that you can move.

We used box2d velocity to set the rate at which it wraps

iBg.setWrap(Texture.TextureWrap.Repeat,Texture.TextureWrap.Repeat);

scrollTimer += body.getLinearVelocity().x/(1000);

if(scrollTimer>1.0f)

scrollTimer = 0.0f;

spBg.setU(scrollTimer);

spBg.setU2((scrollTimer+1));

**All together:**

**Branch: master**

Combined previous concepts into the main project

**Box2d with buttons**

if(bLaunchPressed){

body.setLinearVelocity(20f,10f);

}

**Sidescrolling with box2d velocity**

scrollTimer += body.getLinearVelocity().x/(1000);

**Resetting all aspects when the button is pressed**

body.setLinearVelocity(0, 0);

body.setAngularVelocity(0);

body.setTransform((spPeng.getWidth() + spPeng.getWidth() / 2) / fPM,

(spGround.getHeight() + spPeng.getHeight() / 2) / fPM, 0);

camera.position.y = spGround.getY()+camera.viewportHeight/2;

scrollTimer = 0f;

**Texture Arrays**

**Source:** <http://gamedev.stackexchange.com/questions/80182/how-do-i-initialize-and-use-a-texture-array-libgdx>

**Branch: BackgroundSelection**

Texture[] tile = new Texture[3]; // initialize the array

tile[0] = new Texture(Gdx.files.internal("cave.jpg")); // initialize the image positions in array

tile[1] = new Texture(Gdx.files.internal("Rainforest.jpg"));

tile[2] = new Texture(Gdx.files.internal("city.jpg"));

// use a for loop to loop through the array to

TempImg = tile[0];

for(int i = 0;i<tile.length-1;i++){

tile[i]=tile[i+1];

}

tile[tile.length -1] = TempImg; // create a TempImg which is the image being displayed currently

**Text Input + File IO**

**Branch: InputHandling**

**Source:**

**http://tutorial-libgdx-android.blogspot.ca/2014/02/handling-inputs-text.html**

[**https://github.com/libgdx/libgdx/wiki/File-handling#writing-to-a-file**](https://github.com/libgdx/libgdx/wiki/File-handling#writing-to-a-file)

FileHandle file = Gdx.files.external("file.txt"); // load the file

sInitial = "Your name:";

sDialogue = "Flightless Fury";

afficher = false;

batch = new SpriteBatch();

font = new BitmapFont();

Gdx.input.getTextInput(new TextInputListener() { // add action listener

@Override

public void input(String sName) {

sMessage = "Hello, <"+sName+"> Welcome Flightless Fury ";

file.writeString("welcome",true); // write to the file

afficher =true;

}

This allows us to test out the user progress storage aspect of our game. This way every time a game is launched the user must enter a username to high scores are kept, in an orderly manner.

**Major Challenges/setbacks( reference specific code please):**

**Angles**

Initially, in Angles, we only had the up button working. These two week, we wanted the penguin to keep on rotating whenever the button is pressed rather than just rotating once. we got that working. Whenever the up button is pressed, the penguin keeps on rotating instead of just rotating once. The code below goes in the create method and we found it from this link: <http://gamedev.stackexchange.com/questions/60123/registering-inputlistener-in-libgdx>

UpButton.addListener(new InputListener()

{

@Override

public boolean touchDown(InputEvent event, float x, float y, int pointer, int button)

{

bUpPressed = true;

return true;

}

@Override

public void touchUp (InputEvent event, float x, float y, int pointer, int button)

{

bUpPressed = false;

}

});

Touch Down method is whenever the button is touched and Touch Up method is whenever the button is not pressed. as you can see, we have a boolean which tells us if the buttons is pressed or not. The following code works with the boolean to rotate the penguin up 3 degrees continuously.

if (bUpPressed){

penguin.peng.rotate((float)3);

}

But when we added the down button to the code, the down button worked but the up button stopped working. I tried everything I could, I wrote the down button code before the up button code, it had the symptoms. I had the input listener codes in main initially, I tried putting the input listeners in their dedicated class files, and still it did the exactly same thing. Then, matt suggested me to make a class files with both buttons together, that didn’t work either. We currently don’t know what is wrong with the code, our biggest set back yet

**Input Handling and File IO**

FileHandle handle = Gdx.files.external("myfile.txt");

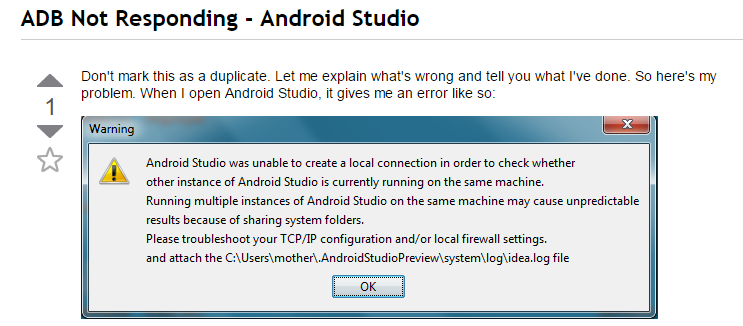
In this case, “myfile.txt” needs to be in the users’ [home directory](http://wikipedia.org/wiki/Home_directory)(/home/<user>/myfile.txt on Linux, /Users/<user>/myfile.txt on OSX and C:\Users\<user>\myfile.txt on Windows) on desktop, and in the root of the SD card on Android.

**Texture Array**

Error Loading image - buttons.png and buttons.pack

errors on initializing array

**Android Studio Error**



Solution: <http://tips.androidhive.info/2013/10/android-stop-or-start-adb-from-command-line/>

Open the cmd from **Start ⇒ run ⇒ cmd** and execute these commands.

**Before you execute the commands in CMD make sure that you added the adb tool to your Environment Variables path.**

## Killing adb

|  |
| --- |
| adb kill-server |

**Gradle Syncing Problems:**

Source: http://stackoverflow.com/questions/21066598/android-studio-0-4-2-gradle-project-sync-failed-error

|  |  |
| --- | --- |
| down voteaccepted | 1. File -> Invalidate caches / Restart 2. Shutdown Android Studio 3. Rename/remove .gradle folder in the user home directory 4. Restart Android Studio let it download all the Gradle stuff it needs 5. Gradle build success ! 6. Rebuild project.... success ! |

**Any modifications to your specifications/release schedule:**

|  |  |
| --- | --- |
| **1.15** | **Add Edge Panning for up and down motion(Background moves only when the penguins reaches the edge)** |

|  |  |
| --- | --- |
| **1.15** | **Orthographic Camera will follow the penguin, while maintaining map position** |

**Description of your scratch/test program:**

Describe the generic concept you needed to test out:

File IO and text Input

Source any web site/book that helped you with that concept:

Source: <http://tutorial-libgdx-android.blogspot.ca/2014/02/handling-inputs-text.html>

<https://github.com/libgdx/libgdx/wiki/File-handling#writing-to-a-file>

<http://gamedev.stackexchange.com/questions/60123/registering-inputlistener-in-libgdx>

Describe the code and the lesson that you learned from it:

That file IO is somewhat simple, but there are its negatives such as specific file location.

Describe any challenges that you enjoyed in integrating this scratch code into your major project:

In this case, “myfile.txt” needs to be in the users’ [home directory](http://wikipedia.org/wiki/Home_directory)(/home/<user>/myfile.txt on Linux, /Users/<user>/myfile.txt on OSX and C:\Users\<user>\myfile.txt on Windows) on desktop, and in the root of the SD card on Android.

We also enjoyed having the penguin rotate continuously. We looked around everywhere for how to do it but we couldn’t find it. Then, Matt showed us a link of how to do it, we were very excited when it worked.