Scratch Descriptions

# Scratch: Gravity Flipping

<https://github.com/TimCatana/gamegravity/tree/scratch-gravityflipping>

Concept: Allowing a character to flip the direction of its y velocity by pressing a button

No sources used, just previous knowledge.

An interesting thing I used was passing a spritebatch to the render function of the character so that it would draw in accordance with the existing camera and viewport

In ScrGame:

batch.begin();

batch.setProjectionMatrix(camera.combined);

chrMain.render(batch);

batch.end();

In Character:

Public void render(Spritebatch batch){

update();

draw(batch);

}

Due to the fact the Character is an extension of the libgdx sprite class we can use the draw(batch); method as well as other useful methods.

# Scratch Obstacles:

<https://github.com/TimCatana/gamegravity/tree/scratch-obstacles>

The objective of this scratch was to create distinct types of obstacles. I started by making a SprObstacle class which is a subclass of the libgdx sprite class. The obstacle's render function updates its position according to a velocity vector and draws it to the ScrGame sprite batch.

public void render(SpriteBatch batch) {

setPosition(getX() + vVel.x, getY() + vVel.y);

draw(batch);

}

It also has an isHIt method using the libgdx built in hit detection

public boolean isHit(Rectangle player) {

if (player.overlaps(getBoundingRectangle())) {

return true;

}

return false;

}

After this I decided to create an obstacle that moved in a sinusoidal motion. This obstacle is a subclass of the SprObstacle class. I named the class ObsFlying. In this class the only new method I added was an update function, this uses the libgdx sin function and my knowledge from advanced functions to create smooth sinusoidal movement across the screen

public void update(){

setY((MathUtils.sin(getX()/1920\*MathUtils.PI2)\*500)+540);

}

Now with the framework of making subclasses from the SprObstacle class I decided to make another type of obstacle one that rotates

ObsSpinning. The obstacle is a sublcass of SprObstacle which is a subclass of Sprite meaning that this class can use any of the sprite methods. SO to make the object spin all I did was use the rotate function in degrees.

public void update(){

rotate(1);

}

After this I wanted the method of rendering obstacles to be as efficient as possible so i used a libgdx array and because all the obstacles originate from SprObstacle they are all the same type in an array. Additionally each obstacle with unique movement uses the same a function with the same name, update which allows for easier implementation in the array. The obstacles are rendered with a for each loop.

for (SprObstacle obstacle : obstacles) {

obstacle.render(batch);

if (obstacle.getX() < 0 - obstacle.getWidth()) {

obstacle.setX(viewport.getWorldWidth());

obstacle.setY(MathUtils.random(FLOOR, CEILING - obstacle.getHeight()));

}

if(obstacle.isHit(chrMain.getBoundingRectangle())){

System.out.println("HIt " + obstacle);

}

}

In this for loop it renders the obstacle as well as sets its coordinates so it continually loops on the screen and checks for hit detection.

# 

# Scratch Animations:

<https://github.com/TimCatana/gamegravity/tree/PlayerAnimationTesting/core/src/com/catani/gamegravity>

This class was to have a function that takes a spritesheet, splits it up into however many columns and rows there are, and have a specific animation array that takes all of the “Frames” created.

void readyFadeAnimation(String SprSheetName, int nColumns, int nRows, int index) {

Texture SpriteSheet = assets.manager.get(SprSheetName, Texture.class);

TextureRegion[][] tmp = TextureRegion.split(SpriteSheet,

SpriteSheet.getWidth() / nColumns, SpriteSheet.getHeight() / nRows);

txtRegAnimationFrames = new TextureRegion[nColumns \* nRows];

for (int i = 0; i < nRows; i++) {

for (int j = 0; j < nColumns; j++) {

txtRegAnimationFrames[index++] = tmp[i][j];

}

}

AnmCreateAnimation = new Animation<TextureRegion>(0.10f, txtRegAnimationFrames);

}

**This part was the easy part. The problem was that I needed a different animation array for every “txtRegAnimationFrames”. So I tried throwing in my own creates animation array**

void readyFadeAnimation(String SprSheetName, int nColumns, int nRows, int index, Animation<TextureRegion> AnmCreateAnimation) {

... code above

AnmCreateAnimation = new Animation<TextureRegion>(0.10f, txtRegAnimationFrames); }

**The problem was that if I already created Animation<TextureRegion> AnmCreateAnimation, then I can’t create a new instance of it within the function. I did some research to see if there was anyway to set the texture region of Animation<TextureRegion> as txtRegAnimationFrames. But I couldn’t find anything. So I had to do a work around, It’s not the greatest work around, but it works.**

void readyFadeAnimation(String SprSheetName, int nColumns, int nRows, int index, Animation<TextureRegion> AnmCreateAnimation) {

... code above

if (SprSheetName.contains("AniBtnPlaySprite.png")) {  
 nImgNum = 1; }

if (SprSheetName.contains("shopbutton.png")) {  
 nImgNum = 2; }

}

**The nImgNum integer Controls what Array holds what Spirtesheet Like so:**

public Animator(Assets \_assets) {

... code

readyFadeAnimation("AniBtnPlaySprite.png", 2, 6, 0);

if (nImgNum == 1) {

AniBtnFade = new Animation(0.10f, txtRegAnimationFrames);

}

readyFadeAnimation("shopbutton.png", 2, 6, 0);

if (nImgNum == 2) {

AniTest = new Animation(0.10f, txtRegAnimationFrames);

}

}

**Now that each Animation array holds its own frames, I can load them into different functions that runs different animations. These are the functions used is the different screen classes as example: animation.draw**

void drawAni(SpriteBatch batch, int x, int y) { // nImgNum = 1

ElapsedTime += Gdx.graphics.getDeltaTime();

batch.draw((TextureRegion) AniBtnFade.getKeyFrame(ElapsedTime, true), x, y);

}

void drawtest(SpriteBatch batch, int x, int y) { // nImgNum = 2

ElapsedTime += Gdx.graphics.getDeltaTime();

batch.draw((TextureRegion) AniTest.getKeyFrame(ElapsedTime, true), x, y);

}

## 

## Scratch Hit Detection:

<https://github.com/TimCatana/gamegravity/tree/scratch-hitdetection/core/src/com/catani/gamegravity>

https://github.com/TimCatana/gamegravity/tree/scratch-hitdetection/core/src/com/catani/gamegravity

The Red square represents the original hitbox (no polygon) of the rotating obstacle. The blue hitbox around the “SUPREME” logo is the accurate Polygon hitbox. You can drag mario around and a exclamation mark will appear when a hit occurs.

For hit detection between the polygon and boudning rectangle I used a method created by StackOverflow user 1337ingDisorder found here: <https://stackoverflow.com/questions/28522313/java-libgdx-how-to-check-polygon-collision-with-rectangle-or-circle>

**private boolean** isCollision(Polygon p, Rectangle r) {

Polygon rPoly = **new** Polygon(**new float**[] { 0, 0, r.**width**, 0, r.**width**,

r.**height**, 0, r.**height** });

rPoly.setPosition(r.**x**, r.**y**);

**if** (Intersector.*overlapConvexPolygons*(rPoly, p))

**return true**;

**return false**;

}

In the scratch I use an exact copy of his method but when implementing into the main branch I reduced it to a more efficient version for our purposes.

For drawing exact hitbox it is important to use the polygon.getTransformedVertices() function. The vertices with transformations applied are ***only*** calculated when this function is called.

Scratch sprcharredo

<https://github.com/TimCatana/gamegravity/tree/sprcharredo/core/src/com/catani/gamegravity>

Using the idea from:

<https://github.com/libgdx/libgdx/wiki/2D-Animation>

This scratch was just to put all of the animations and functions into the sprChar class to make everything more accessible and efficient. I had major problems with this and I explain it much more in the Challenges section. After I created the PrepareAnimation class that I was having trouble with, I used the direction in velocity in order to decide which animation to draw:

***public void render(SpriteBatch batch) {***

***ElapsedTime += Gdx.graphics.getDeltaTime();***

***setX(getX() + vVel.x);***

***setY(getY() + vVel.y);***

***if (vVel.y == -20) {***

***batch.draw((TextureRegion) AnmCreateAnimation[0].getKeyFrame(ElapsedTime, true), getX(), getY());***

***}***

***if (vVel.y == 20) {***

***batch.draw((TextureRegion) AnmCreateAnimation[2].getKeyFrame(ElapsedTime, true), getX(), getY());***

***}***

***}***

Scratch Coin

<https://github.com/TimCatana/gamegravity/tree/Coins/core/src/com/catani/gamegravity>

We decided to have an in game currency. So I decided to make a class that was similar to Ameers SpcObstacle class for the coin. I called it SprCollectables. This class has an image moving to the left constantly using a velocity vector. That is given when you first initialize the object.

***public class SprCollectables extends Sprite {***

***Vector2 vVel;***

***public SprCollectables(String path, float xVel) {***

***super(new Texture(path));***

***vVel = new Vector2(xVel, 0);***

***setPosition(1920 / 2, 1080 / 2);***

***}***

***public void render(SpriteBatch batch) {***

***setPosition(getX() + vVel.x, getY() + vVel.y);***

***draw(batch);}***

***public boolean isHit(Rectangle player){***

***if (player.overlaps(getBoundingRectangle())) {***

***return true;***

***}***

***return false;***

***}***

***}***

After this I made a hit detection function that was similar to the sprObstacles hit functiion. When the player hits, the coin is redrawn off the screen to the right, keeping its initial X velocity, and comes back towards the character.

**In ScrGame:**

***if(sprCoin.getX() < 0 - sprCoin.getWidth()){***

***sprCoin.setX(viewport.getWorldWidth());***

***sprCoin.setY(MathUtils.random(FLOOR, CEILING - sprCoin.getHeight()));***

***}***

***if(sprCoin.isHit(chrMain.getBoundingRectangle())){***

***System.out.println("HIt");***

***sprCoin.setX(viewport.getWorldWidth() + 100);***

***sprCoin.setY(MathUtils.random(FLOOR, CEILING - sprCoin.getHeight()));***

***}***

# Scratch Description: Firebase-Test

<https://github.com/Ameer-Mushani/firebase-test>

In this scratch it displays a number in the center of the screen , if you tap the left side of the screen it subtracts from the number and if you tap the right side of the screen it adds one to the number. This number is connected to the database, so if the numbers changes on one instance of the app it changes on all instances. So both Tim and I can manipulate the number on our phones as well as each other's. The code is relatively simple it reads from the database every from and updates the database value with the touchDown function. This is the function that updates the database value:

public void updateNumtouches(){

GdxFIRDatabase.instance().inReference("/Tapapp").readValue(Datacache.class, new DataCallback<Datacache>() {

@MapConversion(Datacache.class)

@Override

public void onData(Datacache data) {

Datacache dataRec = data;

nTouches = Integer.parseInt(data.getWord());

}

@Override

public void onError(Exception e) {

System.err.println(e);

}

});

}