Catani WIP 3

# Breakthroughs:

Polygon hit detection:

After realizing that the spinning object had an incorrect Hitbox. I scoured the web to find a solution, after much research I came to the conclusion that the best method is using a Libgdx Polygon object as the dynamic hitbox of the spinning rectangle object. This is detailed in the scratch description below.

# Challenges:

I have been having a lot of trouble with the player animation. I took it upon myself to ditch my previously created animator class and put all of the player animations into the SprChar class that Ameer has made. Creating the actual animation was the easy part since I have done it multiple times already. When I ran my first test program, the animation worked fine, but for some reason the velocity vector that we made did not work and the animation would not flip when I tell it to. Here was the problem:

In ScrGame main loop:

***if (chrMain.getY() > CEILING - chrMain.getHeight()) {***

***chrMain.setY(CEILING - chrMain.getHeight());***

***}***

***if (chrMain.getY() < FLOOR) {***

***chrMain.setY(FLOOR);***

***}***

then:

***public boolean touchDown(int screenX, int screenY, int pointer, int button) {***

***if (chrMain.getY() == FLOOR || chrMain.getY() == CEILING - chrMain.getHeight()) {***

***chrMain.flipY();***

***}***

***return true;***

***}***

I know this isn’t the best way to do this because there can be some bugs but this was to get the gist of what was going on. I focused more on what happens when the character is on the floor. So the characters velocity starts at -20. When the character gets to less than floor we set it to floor. So the characters getY() should be floor. In the touch down boolean I set it to if it is equal to floor flipY(). The flipY() function looks like this:

***public void flipY() {***

***flip(false, true);***

***vVel.y \*= -1;***

***}***

This function is in our sprchar class and flips velocity and flips the spritesheet with is set as the super texture. I found out that the flip function just doesn't work and I don't know why. But the velocity function should flip. I spent a full week on trying to the the image to flip. When I did some debugging, I found out that the getY() of the animation was 500 or 600 each time I ran the program, this makes no sense because the FLOOR variable is equal to 169 and we set the sprChar to floor the whole time. The even bigger problem was that the image itself when you look on the screen is visually at the FLOOR. But the program thought it was at 500 - 600 for some reason. **AFTER THIS****the whole main branch stopped functioning and would not compile… So I had to spend a full day in reverting everything I did and fixing the main.**

So I redid everything I previously did in the SprChar class with some minor changes and got rid of some stuff in the ScrMainMenu screen and… It started working… I have no idea why it started working but it works and I’m happy. After this another problem appeared. The image I was using would not flip. So I needed to create two different spritesheets. One with a normal spritesheet and one with the spritesheet flipped. I just took the easy way out and created two different animations, one with the character running right and one with the character running right upside down.

***public void PrepareAnimations(int rows, int columns) {***

***tmp = TextureRegion.split(SpriteSheet, SpriteSheet.getWidth() / columns, SpriteSheet.getHeight() / rows);***

***tmp2 = TextureRegion.split(SpriteSheetInverted, SpriteSheetInverted.getWidth() / columns, SpriteSheetInverted.getHeight() / rows);***

***txtregRunLeft = new TextureRegion[4];***

***txtRegRunRight = new TextureRegion[4];***

***txtRegRunRightInv = new TextureRegion[4];***

***for (int i = 0; i < rows - 1; i++) {***

***for (int j = 0; j < columns; j++) {***

***txtregRunLeft[index++] = tmp[i][j];***

***}***

***}***

***index = 0;***

***for (int i = 1; i < rows; i++) {***

***for (int j = 0; j < columns; j++) {***

***txtRegRunRight[index++] = tmp[i][j];}***

***}***

***index = 0;***

***for (int i = 0; i < rows - 1; i++) {***

***for (int j = 0; j < columns; j++) {***

***txtRegRunRightInv[index++] = tmp2[i][j];***

***}***

***}***

***AnmCreateAnimation = new Animation[3];***

***AnmCreateAnimation[0] = new Animation<TextureRegion>(0.10f, txtregRunLeft); // right***

***AnmCreateAnimation[1] = new Animation<TextureRegion>(0.10f, txtRegRunRight); // left***

***AnmCreateAnimation[2] = new Animation<TextureRegion>(0.10f, txtRegRunRightInv); // right inverted***

***}***

## Description of Scratch Hit Detection:

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

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

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()));***

***}***

# Updated Release Schedule:

|  |  |
| --- | --- |
| **Release Name** | **New incremental features of this release** |
| **1.0** | **Scrolling background and ability to flip gravity** |
| **2.0** | **Scrolling obstacles with hit detection** |
| **3.0** | **Add coin with random spawn locations** |
| **4.0** | **Develop Main Menu and Gameover screen that transition into one another as well as Game screen** |
| **4.5** | **Add dynamic score in meters and speed up scrolling background as game progresses** |
| **5.0** | **Add larger variety of obstacles such as pitfalls and spawn obstacles with increasing difficulty for the player as score progresses** |
| **6.0** | **Change terrain, remove flat ground. Replace with different sections of level (Example: Rolling hills, stairs going up/down etc.** |
| **7.0** | **Add shop with variety of skins for main character** |
| **8.0** | **Add sounds (flipping sound, death sound etc.)** |
| **9.0** | **Attempt firebase multiplayer on android devices** |