<https://github.com/spencersmith24/CS315-Godot-Exercises>

# Controlling animations on player objects

A screenshot of a video game

Description automatically generated

A screen shot of a computer code

Description automatically generatedControlling animations on player objects is not hard at all. First, you need an animation to control, which, in this case, is made using an AnimatedSprite2D node. The frames of the animation are assigned, and the animation is made. To actually play the animation in the game, it needs to be started. This is done in the code using the function ***play()***. Once this line of code is called, the specified animation will be played.

# Structuring and using sound effects effectively

A screen shot of a computer screen

Description automatically generatedA screenshot of a phone

Description automatically generated Structuring sound effects doesn’t have to be hard. In this case, we simply created a 2DNode for them to be stored in and referenced the node when it was time for the sound to be played. A sound effect can be played just like an animation, by using the ***play()*** function. To ensure that the sounds aren’t played when they don’t need to be, **$SFX/Thud.play()** is called in the **plane\_hit()** function. This means that the sound will only be played when the plane runs into a rock.

# Building and using a user interface GUI

A screenshot of a computer

Description automatically generatedBuilding a GUI is important because it allows the player to understand what is going on. In this game, the only GUI we have is a “game over, try again?” one, that tells the player they lost. This case, it’s only made up of two components: a label and a button. Even though the game over screen is very important here, players don’t want to see it all the time, so it’s hidden by default. The process of unhiding it and presenting it to the player is done with an animation, which I’ll talk about further down, but for now let’s assume the player has lost. This is what they will see:

A grey background with white text

Description automatically generated

A computer screen shot of a code

Description automatically generated

Once the Play Again button is pressed, the function **\_on\_button\_pressed()** runs. This function starts the rock spawning timer back up, makes the plane visible again, begins processing the plane again, and hide the Game Over screen.

# Difficulties of resetting games

For me, the hardest part of resetting the game was getting the plane to go back to it’s starting position. For some reason, it moves back to the correct position, but then when it “flaps,” it clips back to the position it was at when it died. I’ve spent quite a bit of time trying to figure it out, but I just have not been able to. Generally speaking, however, difficulties resetting this game could come from stopping and starting the spawning of certain objects, stopping the game from listening to inputs, etc.

# Using animations for game "feel"

A screenshot of a video editing software

Description automatically generated A screenshot of a video editing software

Description automatically generated

In this game, when you lose, the game over screen gracefully slides down from the top of the screen to the middle. So graceful, in fact, that you don’t even feel bad about losing. Without the use of animations, the game would feel clunky. Instead of a pretty slide down, it would just pop up on top of the screen, which is NOT graceful. The other animation in this game is the plane’s propeller spinning, which makes the character feel more alive.

# Conversation

Evan helped me figure out the sprite sheet to get the plane animation to work. I knew I needed to use the sprite sheet to get the animation frames, but I couldn’t remember *how*. Thankfully, Evan was able to explain it to me and I got it working.