**Welcome to Day #4 of CGCC!**

Every day we will have a GitHub repository page that outlines each day and the activities that we will complete. We will also provide all homework on these pages.

Feel free to browse the other days to see what is coming up!

As always, let us know if you need any help or have any questions.

*Link to Camp GitHub*: <https://github.com/paigerodeghero/ClemsonGameCodingCamp/tree/master/2021>

**Day 4: Game Narrative Development**

**SCHEDULE:**

* Instructors start video call and recording
* Reviewing work from Day 3
* What is Prototyping + video
* Application Prototyping
* Final Project Brainstorming
* Find some basic assets
* Final Project Prototyping

**INSTRUCTION**: Reviewing yesterday’s scene/character creation (approximately 5 minutes)

Have each student group review each other's scenes and characters (2-3 minutes each)

**INSTRUCTION**: What is prototyping? (approximately 10 minutes)

* Game prototyping is an important component in the game development process. This involves creating a method to test the concept of the game, to see if the idea of the video game can be put into practice before investing too much money and time in the project. (<https://starloopstudios.com/rapid-game-prototyping-why-is-it-important-in-game-development/#:~:text=Game%20prototyping%20is%20an%20important,and%20time%20in%20the%20project>.)
* Discuss why it is important we try to prototype games before development starts. (<https://www.uxpin.com/studio/blog/paper-prototyping-the-practical-beginners-guide/>)
  + Rapid iteration
  + Inexpensive
  + Increased creativity
  + Less of an up-front learning curve
  + Etc.
* Different ways to prototype:
  + Paper
  + Digital
  + Native
  + Any others or ‘subdivisions’ of the above methods (like sticky notes)?

**INSTRUCTION:** Summarize with a prototype video (7.5 minutes)

* <https://www.youtube.com/watch?v=JMjozqJS44M>

**ACTIVITY:** Create a paper prototype of a common application such as Spotify or YouTube (~20 min) (pairs)

* Draw this application with marker/paper/scissors/ whatever you feel is necessary.
* List all the functions of this application and how each element of your design is used to do this function (A play button plays the music)
* Take a picture of your design and upload it to (#WHERE\_TO\_UPLOAD)
* Share with your partner why you chose to include certain elements. Where there any that you decided to leave out? Why?

Discuss what having too many features can do to an application. Where do we draw the line on what is a good, useful feature and what is meaningless clutter?

10 Minute Break

**ACTIVITY:** Brainstorm Final Project Game (~40 min)

* Characters: (10 min)
  + The Fighter: A hero who fights the enemy with their fists, feet, or weapon.
    - Examples: Street Fighter, Karateka, Mortal Kombat
  + The Big Bad: A dastardly villian who appears to be stronger than the hero, but has a hidden weakness.
    - Examples: Gannon, Donkey Kong
  + The Sage: A mystic, an odd character allied with the hero who provides helpful hints at crucial times during the game.
    - Examples: Legon of Zelda, Skyrim, Metal Gear Solid
  + The Sidekick: A companion to the hero who provides comic relief or aids the hero in solving their quest.
    - Examples: Luigi in Mario Bros, Yoshi in Mario World, Sonic 2's Tails.
* Narrative: (15 min)
  + Overcoming the monster: The hero must flight and slay the monster that threatens their community.
    - Examples: Beowulf, Dracula, King Kong, Pacman, Mario Bros., Space Invaders, Asteroids, Galaga
  + Rags to Riches: An insignificant person is dismissed by others. Something happens to elevate them, revelaing that person to be exceptional.
    - Examples: Ugly Duckling, Aladdin, Superman
  + The Quest: The hero must set out on a long hazardous journey to battle obstacles until they are triumphant.
    - Examples: Lord of the Rings, Harry Potter, Wizard of Oz
  + Voyage and Return: The hero travels out of their normal world into the unknown and overwhelming, before escaping back to the safety of their home.
    - Examples: Alice in Wonderland, Finding Nemo, Gulliver's Travels, Legend of Zelda, Super Mario Bros.
  + Rebirth: The hero falls under a dark spell (e.g. sleep, sickness, enchantment) before breaking free and being redeemed.
    - Examples: Sleeping Beauty, Beauty and the Beast
  + The Neverending Story: A repetitive story with infinite challenges that get more and more difficult to beat
    - Examples: Donkey Kong, Q\*bert, Tetris
* Rules: Defines how the characters can move through the game world and describes the actions they can take and their effects. (15 min.)
  + Navigation
    - Walking, Running, Swimming, Flying
    - Constraints to only walk up/down, left/right
  + Information
    - Reading a scroll
    - Listening to a character
  + Inventory
    - Picking up an item
    - Choosing to use an item
    - Dropping an item
    - Losing an item
  + Obstacles
    - Jumping
    - Running through
    - Punching at
  + Fighting
    - Punching at
    - Jumping on top of
    - Kicking
    - Running through
    - Round-off back handspring
  + Dying
    - Getting run over
    - Getting hit
    - Jumping into a pit
    - Running into yourself
  + Winning
    - Eating all the food
    - Defeating all the enemies
    - Solving all the puzzles
* Technology:
  + Mapping the game buttons to player actions
  + Secret game modes
    - Example: Pressing A and B buttons together makes the character invisible to monsters.

**ACTIVITY:** Finding some basic assets

(~30 min total, ~25 min for searching, ~5 min for GitHub)

* Now that we have a basic idea of what the game is going to be, we’ll start looking for assets to prototype your final project idea.
* Linked are several different free and Creative Commons / public domain asset libraries that will make it easy to just search, download, and add different assets.
  + <https://opengameart.org/>
  + <https://itch.io/game-assets/free> (itch does have premium assets that need to be paid for, but still have thousands of good free ones)
  + <https://www.gamedevmarket.net/category/2d/?type=free> (basically the same as itch.io)
  + [Kenney.nl](https://www.kenney.nl/)
* Have one person share their screen and create a folder labeled Assets in your GitHub Repository
  + Within the Assets folder, create subfolders for Characters, Scenery, and Weapons
* The person sharing the screen then navigates to Asset libraries and everyone decides on which assets they want to include in the game
  + Screen sharing teammate downloads the assets agreed by the entire group and saves them into the GitHub folders
    - Continue until all assets are found or time is up
  + Once complete, the screen sharing teammate pushes the changes to GitHub and the other team members can then pull the changes onto their local computers

10 Minute Break

**ACTIVITY:** GoDot Development: Adding Features to Games (~40 min)

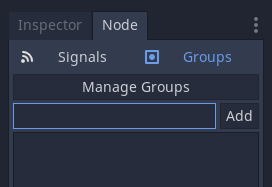
Go ahead and download the SpaceRocks\_Unfinished\_NoExplosion folder from GitHub. [*More explicit instructions for students are needed for GH download*]

This is a partially finished game, go ahead and run it. Move the space ship forward with W or the up arrow, turn the space ship left with A or left arrow, turn the space ship right with D or right arrow, and shoot laser beams with the spacebar.

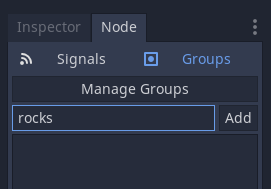
Try shooting a rock. What does it do? It does nothing, and we’re going to fix that in this activity.

Take a minute or two to get accustomed to the code and the file structure.

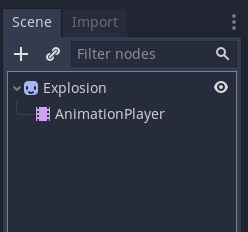
1. So now we need to Explode the Rocks in the scene by shooting them. The first step we need to do is create a rocks group and create the explosion animation.
   1. Start by going to the Rock.tcsn in the GoDot editor. Next to the Inspector, you should see a Node tab. Click on that and then switch to the Groups tab right underneath it.



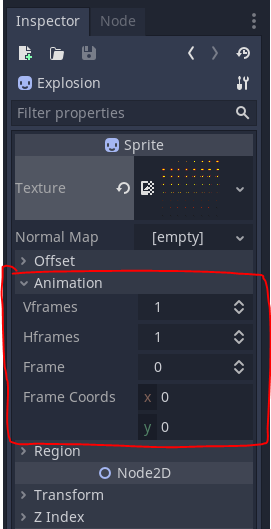
* 1. Next, we need to add a rocks group to our scene. In the text box, type rocks and click add.



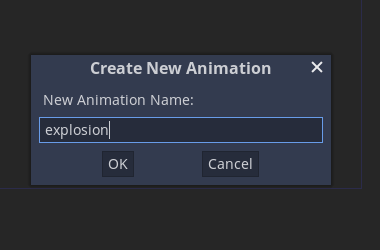
* 1. Save your Rock.tcsn scene (Ctrl + S) and create a new scene. This new scene will be for the explosion animation. We need to add a Sprite (renamed to Explosion) and an AnimationPlayer as the child of the Sprite.



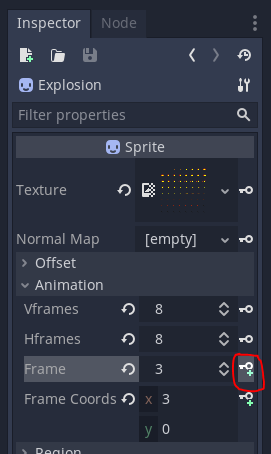
* 1. We need to add a texture to the Sprite, so go to the Inspector and add explosion.png as the Texture. Since this texture is made up of multiple images made to be an animation, we need to do a little more processing on the image.
     1. In the Inspector, we need to go to the Animation tab.



* + 1. Set the Vframes and the Hframes to 8. This will slice the image into smaller images that we will then stitch together to make the explosion animation.
  1. Now we need to add a new Animation. Click on the AnimationPlayer node and go to the animation panel at the bottom of the GoDot window.
  2. Add a new Animation and name it explosion

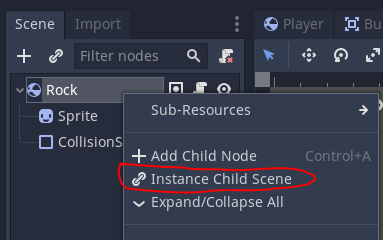


* 1. Update the Animation Length to 0.65 seconds and the Step to 0.01
  2. Now click on the Sprite (Explosion) Node and click on the Key button next to the Frames in the Inspector. Click on the key button until you reach the end of the animation.



* 1. Save your Explosion scene! Now click on the Play button in the Animation Editor tab to watch the explosion!

1. Now we need to add our Explosion scene instance to our Rock Scene. Go to Rock.tcsn in the GoDot editor.
   1. Right click on the Rock node and click “Instance Child Scene” and choose our explosion scene.



* 1. Now we need to add this line of code to the end of the start() function in Rock.gd:

$Explosion.scale = Vector2(0.75, 0.75) \* size

* 1. In our variables at the top of the script, we need to add a signal, like this:

signal exploded()

* 1. Now we need to add an explode() function:

func explode():

    layers = 0

    $Sprite.hide()

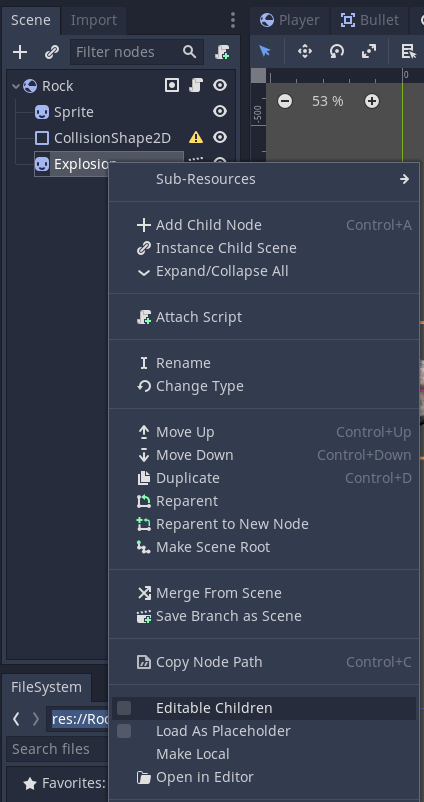
    $Explosion/AnimationPlayer.play("explosion")

    emit\_signal("exploded", size, radius, position, linear\_velocity)

    linear\_velocity = Vector2()

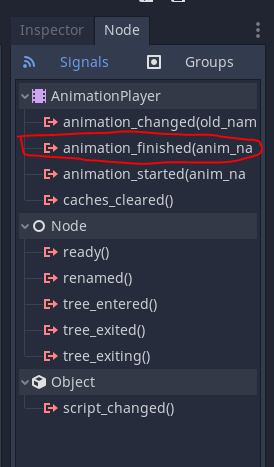
    angular\_velocity = 0

1. Once the animation finishes playing, the AnimationPlayer will emit a signal, but first we need to make the AnimationPlayer visible within the Rock scene.
   1. Right click on the Explosion in the Rock scene.



* 1. Check the Editable Children Box to make the AnimationPlayer viewable.

1. Now we need to connect the AnimationPlayer’s animation\_finished signal to the Rock’s script to let the rock know when the animation is finished.



* 1. Double-click the animation\_finished signal, make sure you’re connecting it to the Rock scene, and click connect.
  2. Then, in the Rock.gd script, we need to add one line to the function that was just added.

func \_on\_AnimationPlayer\_animation\_finished(anim\_name):

    queue\_free()

* 1. Now make sure to save your script and play your game, you should be able to shoot the rocks and watch them explode!

1. For an additional challenge, let’s make our rocks spawn smaller rocks. That way we have more rocks to shoot at in each level.
   1. In our Main.gd script, we need to add this line to the end of our spawn\_rock() function:

r.connect('exploded', self, 'on\_Rock\_exploded')

* + 1. This line connects our rock’s exploded signal to a function in our Main script called on\_Rock\_exploded
  1. So now, we need to create our on\_Rock\_exploded() function as shown below:

func on\_Rock\_exploded(size, radius, pos, vel):

    if size <= 1:

        return

    for offset in [-1, 1]:

        var dir = (pos - $Player.position).normalized().tangent() \* offset

        var newpos = pos + dir \* radius

        var newvel = dir \* vel.length() \* 1.1

        spawn\_rock(size-1, newpos, newvel)

* 1. Save your script and run your game! When you shoot rocks, there should be two more smaller ones spawning that are moving in two different directions!