**Welcome to Day #2 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>

**Links for Camp Day #2**:

* GoDot Engine
  + <https://godotengine.org/>
* GitHub Overview Video
  + <https://www.youtube.com/watch?v=w3jLJU7DT5E&t=30s>
* GitHub Classroom
  + TODO
* How to use GitHub?
  + <https://docs.github.com/en/github/authenticating-to-github/creating-a-personal-access-token>
  + <https://github.com/git-guides/>
* Game engines
  + <https://youtu.be/DKrdLKetBZE>
* FlappyBird
  + <https://github.com/paigerodeghero/ComputerGameCodingCamp/blob/755d6677c9736b0f93f97e8be910aa04144559ec/2021/godotexamples/FlappyBird.zip>

**Optional Resources**:

* What is open source software (OSS)?
  + <https://www.youtube.com/watch?v=a8fHgx9mE5U>
* Tutorials:
  + FlappyBird
    - <https://youtu.be/8_ThGJG9Kqg>

**Day 2: GitHub, Godot, and creating first game**

**SCHEDULE**:

* Homework review
* Meeting your team
* GitHub Classroom setup
* GitHub introduction
* “FlappyBird” story
* Godot game engine
* Making “FlappyBird”
* Homework: Make one change to “FlappyBird” and commit to GitHub

**ACTIVITY**: Homework Review (20 minutes)

Homework Review:

* Each student presents their findings from the game they choose to play
* Discuss the following game elements from the game
  + Goal
  + Story
  + Rules
  + Players
  + Player interactions

**ACTIVITY**: Meeting your team (30 min)

* Students are introduced to their virtual workspace
* Activity: Meeting your team.
  + Introductory Interview with Project Partner
    - Name
    - Which grade are you in?
    - What is your background with computers? What skills do you have?
    - What do you find most interesting about computers and how does this impact your college plans?
    - How do you manage your time when you get busy with a lot of tasks?
    - Have you worked on a team project before? If yes,
    - How often did your team meet together?
    - Did your team have a leader? What did that leader do?
    - What was your role on the team?
    - How well did you get along with your teammates related to work, or related to non-work?
    - Who are the other members of your family? Do you live with them?
    - What are your hobbies/interests/passions that are not related to this class?
    - Do you have any unique skills/tricks that you can show me now?
    - Who is your favorite teacher in high school? What do they teach? Why are they your favorite?
    - What was the last non-software-related book you read and describe what it was about in 2 sentences.
    - What is the best movie you saw in 2020, and why was it your favorite?
    - Do you have a personal hero in your life? Who is it and why are they your hero?
    - What is your phone number? For backup purposes in case technology doesn't work.
    - When can you work on the project outside of class?
    - Negotiate mutually exclusive times to edit code
    - What kind of programming experiences have you had in the last 3 years?
    - Tell me about a trip you took that was far away from where you live?

**ACTIVITY**: GitHub Classroom setup (20 min)

* Invite students to GitHub classroom
* Students accept the invitation
* Make sure all student groups and projects are setup in GitHub

**INSTRUCTION**: GitHub Introduction (5 min) (Put it in context with game development)

* GitHub intro
  + What is GitHub?
    - <https://www.youtube.com/watch?v=w3jLJU7DT5E&t=30s> (check)
  + Create a token
    - <https://docs.github.com/en/github/authenticating-to-github/creating-a-personal-access-token>
  + How to use GitHub?
    - <https://github.com/git-guides/>
    - Commit
    - Pull
    - Push
    - Merge

**INSTRUCTION**: Creating a game: Story (10 min)

Who is Faby?

* Discuss the story of flappy bird
* What is the goal?
  + To go through as many obstacles as possible
* How can we make the game more interesting?
  + Add enemies in the game

**ACTIVITY**:How to make FlappyBird more interesting? (5 min)

* Students share their ideas to make FlappyBird more interesting

**INSTRUCTION**: INTRO GODOT: (game engines in general) (5 min)

* What is a game engine?
  + A game engine is a framework that allows us to create games without worry about how our code interacts with the computer
  + <https://youtu.be/DKrdLKetBZE>
* What is godot?
  + Is an open-source game engine
* Games made in godot
  + Go over a few and/or show
    - The Adventures of Dog Mendonça & Pizzaboy
    - Commander Keen in Keen Dreams

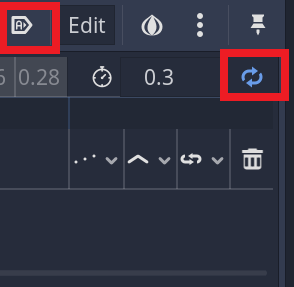
**INSTRUCTION**: Setup Visual studio code (20 min)

* Download visual studio code
  + Open visual studio code
    - Open extensions
    - Install the following
      * Godot-tools
      * Live Share Extension Pack

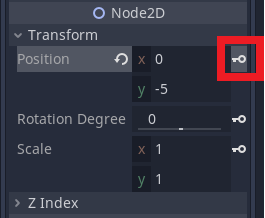
15 min break

**ACTIVITY**: Create “FlappyBird” in godot [see one – do one – show one]

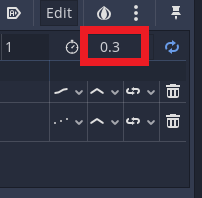
* **INSTRUCTION**: Download project starter from GitHub (10 min)
  + Use this link to download the starter version of FlappyBird
    - <https://github.com/paigerodeghero/ComputerGameCodingCamp/blob/755d6677c9736b0f93f97e8be910aa04144559ec/2021/godotexamples/FlappyBird.zip>
  + Unzip the file
* **INSTRUCTION**: Open the project (5 min)
  + Open godot and select **Import**
  + Navigate to the FlappyBird project and open the folder
  + Click on the **project.godot** file
  + Select **open** and then **import & edit**
* **INSTRUCTION**: Open the project in visual studio code, press F1 and select the below option (5 min)
  + - Godot Tools: Open workspace with Godot editor
    - Point it towards you Godot installation
* **INSTRUCTION**: Create player object (10 min)
  + Add a **2d node** by clicking on it
  + Rename it as World
  + Save the scene
  + Create a new scene by using the **+** next to World
  + Press **+** to add a **RigidBody2D** node. Rename the node to player
  + Press **+** to add a **Sprite**
  + Press **+** to add a **CollisionShape2D**
  + Select the **Sprite**. Drag the **bird.png** file into **Texture** in the **Inspector**
  + In **Inspector** -> **Animation** -> **Hframes**, set the value to 3
  + Select **CollisionShape2D**. In **Inspector** -> **Shape**. Select **New CircleShape 2D**
  + Increase the size of the collider as required
* **INSTRUCTION**: Create player animation (20 min)
  + Use **+** to add an **AnimationPlayer**
  + Create idle animation
    - Use **Animation** -> **New** to create a new animation
    - Name it as **idle** and press **OK**
    - Enable auto play and looping using the buttons



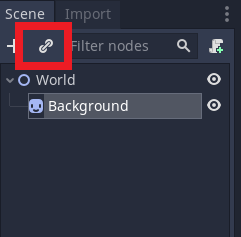
* + - Select **Sprite** from **Scene** tab and navigate to **Transform** in **Inspector**
    - Change **Transform** -> **y** to **-5**
    - Create a keyframe by pressing the key button



* + - Change **Transform** -> **y** to **5**
    - Select time **0.5** on animation
    - Create another keyframe
    - Set interpolation mode to **cubic** for a smoother animation
  + Create flap animation
    - Use **Animation** -> **New** to create a new animation
    - Name it as **flap** and press **OK**
    - Enable looping using the button
    - Select **Sprite** from **Scene** tab and navigate to **Transform** in **Inspector**
    - Change **Transform** -> **y** to **0**
    - At time 0, create a keyframe by pressing the key button
    - Using the key button next to **Frame**, create keyframs for all 3 frames
      * Create first keyframe at 0, second one at 0.1 and third one at 0.2
    - Set duration of animation to **0.3**



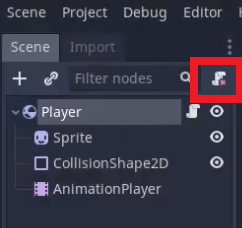
* **INSTRUCTION**: Add player instance to World scene (5 min)
  + Open the World scene
  + Create an instance of the Player



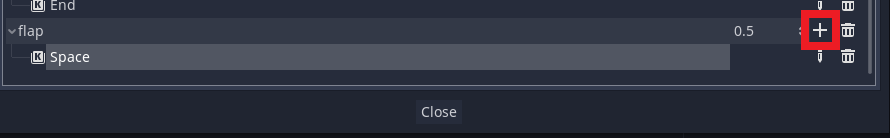
* + Place player in the middle of the scene
    - You can use the mouse to click and drag the player or you could use the transform property
  + Save the scene and name it player

15 min break

* **INSTRUCTION**: Make player move with keyboard input (45 min) (VS code liveshare)
  + Open **Player** scene
  + Select **Player**
  + Set **Inspector** -> **RigidBody2D** -> **Gravity Scale** to 0
  + Add a new empty script to the player by using the script button



* + Define an action
    - Navigate to **Project** -> **Project Settings** -> **Input Map**
    - Add an action **flap**
    - Use the **+** to link action with space bar press
      * Key -> Press on Space -> OK



* + Save all scenes
  + Add the below code to the script
    - This will move the player up and down as well as play the flap animation

extends RigidBody2D

var started = false

func \_physics\_process(delta):

    if Input.is\_action\_just\_pressed("flap"):

        if !started:

            start\_flap()

        flap()

func start\_flap():

    if started: return

    started = true

    gravity\_scale = 5.0

    $AnimationPlayer.play("flap")

func flap():

    linear\_velocity.y = -200

* + Add rotation to player on button press

extends RigidBody2D

var started = false

func \_physics\_process(delta):

    if Input.is\_action\_just\_pressed("flap"):

        if !started:

            start\_flap()

        flap()

    if rotation\_degrees <= -30.0:

        angular\_velocity = 0

        rotation\_degrees = -30.0

    if linear\_velocity.y > 0:

        angular\_velocity = 3.0

        if rotation\_degrees >= 90:

            angular\_velocity = 0.0

func start\_flap():

    if started: return

    started = true

    gravity\_scale = 5.0

    $AnimationPlayer.play("flap")

func flap():

    linear\_velocity.y = -200

    angular\_velocity = -8.0

* **INSTRUCTION**: Add limits to where the bird can go (5 min)
  + Modify Player.gd script to sop the player from falling beyond a limit

extends RigidBody2D

var started = false

func \_physics\_process(delta):

    if Input.is\_action\_just\_pressed("flap"):

        if !started:

            start\_flap()

        flap()

    if rotation\_degrees <= -30.0:

        angular\_velocity = 0

        rotation\_degrees = -30.0

    if linear\_velocity.y > 0:

        angular\_velocity = 3.0

        if rotation\_degrees >= 90:

            angular\_velocity = 0.0

    if position.y >= 500:

        angular\_velocity = 0

        gravity\_scale = 0.0

        linear\_velocity.y = 0

        $AnimationPlayer.stop()

func start\_flap():

    if started: return

    started = true

    gravity\_scale = 5.0

    $AnimationPlayer.play("flap")

func flap():

    linear\_velocity.y = -200

    angular\_velocity = -8.0

* Add a button to reset the game

**HOMEWORK**:

* Complete all steps shown today
* Make one change to “FlappyBird”
* Commit homework to GitHub