|  |
| --- |
| Chaos Wars Inc. |
| FigTheFrog |
| An exploration of frogger based games. |

|  |
| --- |
| Butcher, Trent Lee  12-30-2023 |

**Objective**

Create a game based on the principles of frogger. Ideally this should be at least a little unique and not a one-to-one copy of the original game.

**Scope**

This section is intended to describe the expectations of the game. This includes but is not limited to the core mechanics, general game flow, UI elements and more.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tractability ID | Description | Source | Confirmation Parameters |  |
| 001 | At least one player must be able to play the game. | Self | One player is playable in game. |  |
| 002 | The game must have a discernable win condition. | Self | Getting to the top position of the map locks in a player life in position. |  |
| 003 | The game map should be bottom to top | Self | The player starts at the bottom and must travel up to the top win positions. |  |
| 004 | Obstacles exist that will result in a lose condition | Self | There may be moving logs/lily pad that allow player to walk over water. Cars kill the player. |  |
| 005 | Movement should be discrete, resulting in a cooldown each movement. | Self | Each time a directional key will result in one movement unit. Holding keys doesn’t move the player. |  |

**Audience**

This section attempts to establish the expected player base. While this is simply a game that attempts to help the developer learn how to use the Godot Engine. That being said, the player base is at least one individual player that can play.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Time Start | Time Stop | Total Time | Notes |
| 12/30/23 | 5:00 AM | 5:30 AM | 0.5 | Finished the preamble and set up workspace. |
| 12/31/23 | 3:30 AM | 8:00 AM | 5.5 | Adde player controller and got it working. |
| 1/2/24 | 7:30 AM | 12:30 PM | 4 | Did some work with the player and speed boost |
| 1/2/24 | 2:00 PM | 5:00 PM | 3 | Updated map and added cars |
| 1/4/24 | 11:00 AM | 2:00 PM | 3 | Added river and Lili-pads |
| 1/4/24 | 3:00 PM | 5:30 PM | 2.5 | Wrapped up adding winCon and enclose map |
|  |  |  | 18.5 |  |

**Log**

12/30/2023 – Today I Completed the preamble of the writeup. Ultimately it will need a few updates however the main foundation for this document is completed.

12/31/2023 – Today I was able to get the Character put together. I found a free frog model that already has a jump animation included. Currently the player will move in discrete steps for each button press. I did this to match the feel of the old frogger gameplay were holding down a directional key won’t see the player move over and over. This method of movement also makes it so the character will only be able to move in one direction at a time meaning that each movement commits the player to a direction. An issue still persists that when a player object collides with a static body, The player will kind of break because it keeps trying to move in whichever direction it wants until it meets a certain distance from the original movement position.

I made a speed boost object that will allow the player to increase their jump distance and speed for a duration. The problem is I am having issues getting the sprite to always be visible to the screen sometimes the angle isn’t right. The sprite was based off an image online, I made it with Aespirte. I am also having issues getting the speed boost to tell the player to increase their speed.

1/2/2024 – Today I did a bit of work getting the speed boost to work. I changed It so that instead of trying to use signals to tell the player to speed up, I just have each object work in tandem. The speed boost sees a player in its collision shape and will delete itself. The player will see the speed boost in its collision shape and will call a “boost” function. This function will increase the speed and jump distance of the player for 5 seconds, might change it but 5 seconds feels fine. I added a particle effect to tell the player they are speed boosted.

I spiced up the map a little adding light post, a grass-colored ground, and a road-colored ground. I will need to add a river and then add some spawners.

The game now has a car spawner that will pseudo-randomly spawn the cars. The cars will travel along for 20 seconds and then de-spawns itself. The cars make a sound as they travel. If the player collides with a car, then the player makes a splat sound and travels with the car for a few seconds before restarting the world. A few errors still persist with cars for starters when a car hits the player if another car hits the player afterward then this can result in a repetitive attempt to end the game meaning that the timer doesn’t finally tick down to zero to restart the map. I think I just need to restructure the kill function to fix this. The car spawner also doesn’t seem to make sure a car is not already spawned and will sometimes spawn cars on top of each other resulting in a janky interaction.

1/4/2024 – Today a lot of progress was made. The Lili pads are now added along with a river that will result in a sink event when you fall in. This was done by having the river collision box toggle a variable in the character object. If there is not a lilipad under the player (being collided with) the player will sink into the river. The lilipad issue was problematic at first as just using a Boolean for “isOnLili” was causing issues when trying to jump from lili to lili. This was fixed by simply adding each unique lilipad to a dictionary when it is encountered and iterating the value “isOnLili” which was switched to an integer. Each time a Lili was jumped off of the object was removed from the dictionary and the isOnLili was reduced accordingly. This works flawlessly and I haven’t been able to get it to break yet.

Rocks were added to either side of the river to serve not only as a wall to inhibit the player from exiting the map but also will sink the player if they run into them. ‘Fires’ were added to either side of the road to inhibit the player’s ability to just jump out of the map. These fires also gave a little more opportunity to play with the particle effects which will come in handy in future projects.

The cars and Lili pads spawners were modified to ensure they don’t spawn on top of each other. A chance roll is now done each time the spawner timer is up to determine if the spawner will actually spawn their respective item.

The issue of colliding with objects and not being able to control the character if it is a static object. To fix this issue a rayCast3d was added to the front face of the ‘frog’ model. This trace would check for solid objects in front of the frog and would toggle the “moving” bool to false if it were the case. This is really nice as it allowed for solid walls (bushes) to be added to enclose most of the map.

Lastly the star was added to yield a win condition for the player. The star will change the active player camera to a more cinematic view that rotates the player. The star itself will rotate and jump up and down to be that much more visible to the player. At the start of the game the player’s camera will start facing low to the horizon before transitioning to the angle that the game is played in. This is done because it was a little ambiguous where the star was to win the game and making this small change to the game made sure the player could see the star.

At this point a few things could be added like a little more lighting, more car models and maybe a little more to touch it up. This could also be paired with a full Ui and menu system. Most of the addons would be a little extra and would need a bit more work, considering nothing to these effects are in the original trackability I am going to elect to add any of those features. Given this the Frogger clone FigTheFrog can be deemed completed.

**Appendix A**

**Synopsis**

The game is now considered mostly completed. While it would have been nice to wrap the game up in an elegant bow with previously mentioned features, they are not located in the original trackability.

Traceability 001 is very clearly met. One player is able to control the frog player and move to avoid obstacles.

Traceability 002 is met partially. I didn’t really want to add the multiple lives functionality as easy as it would’ve been, it would’ve been more work than its worth.

Traceability 003 is met by inspection. The player starts at the bottom of the map and is meant to transition upward.

Traceability 004 is met. Cars spawn on the road in a semirandom rate. The cars will move along the road and if collided with the player it will push the player and play a death sound. I didn’t add a log to jump on while in the river although it would’ve been a matter of finding a model. I was unable to find one however after a short search and didn’t want to make a custom one, so I just skipped that. The cars do spawn as different colors though which is cool. The river will drown, or ‘sink’, the player if they are not standing on a Lili pad while crossing it.

Traceability 005 is met by inspection. If the player presses a directional arrow, they are committed to a movement which lends itself to the feeling of the original arcade game which is cool. Subsequently holding the arrow keys also does not result in continues movement of the player which is specifically called out in the traceability. A power up was also added to allow for the player to speed up their movement two times the original movement for five seconds. A particle effect is emitted from the frog when they are powered up to give player indication of when the powerup wares off. I just added this as a proof of concept even though it wasn’t in the original traceability.

**Resources**

* N/A

**Appendix B**

This section is intended to serve as a reference for the scripts used in the final representation of this project.

“car.gd”

|  |
| --- |
| extends CharacterBody3D  @onready var part1 = $MeshInstance3D  @onready var part2 = $MeshInstance3D2  @onready var part3 = $MeshInstance3D3  @onready var audio\_stream\_player\_3d = $AudioStreamPlayer3D  var SPEED = 2.5  var direction = Vector3(0,0,-1)  var onLili : bool = false  func \_ready():  var parts = [part1,part2,part3]  var colors = [Color.RED, Color.ROYAL\_BLUE, Color.WEB\_GREEN]  var color = colors[randi()%colors.size()]    for part in parts:  if part.material\_override is StandardMaterial3D:  part.material\_override.albedo\_color = color  else:  var new\_material = StandardMaterial3D.new()  new\_material.albedo\_color = color  part.material\_override = new\_material  func \_process(delta):  var face = global\_transform.basis.x.normalized() \* -1  velocity = face \* SPEED  move\_and\_slide()    func \_on\_audio\_stream\_player\_3d\_finished():  audio\_stream\_player\_3d.play()  func \_on\_timer\_timeout():  queue\_free() |

“car\_spawner.gd”

|  |
| --- |
| extends Node3D  @onready var collision\_shape\_3d = $Area3D  @onready var timer = $Timer  var carSlow = preload("res://Scenes/car.tscn")  var rng = RandomNumberGenerator.new()  func \_ready():  timer.start()  spawn()  func spawn():  var check = collision\_shape\_3d.get\_overlapping\_bodies()  var cars = [carSlow]  var car = cars[randi()%cars.size()]  var time = rng.randi\_range(50, 120)/10  timer.wait\_time = time    if check.size() <= 0:  var chance = rng.randi\_range(1, 10)  if chance%3:  var spawnedCar = car.instantiate()  add\_child(spawnedCar)  func \_on\_timer\_timeout():  spawn() |

“FigControler.gd”

|  |
| --- |
| extends CharacterBody3D  @onready var frog = $Frog  @onready var anim = frog.get\_node("AnimationPlayer")  @onready var audioPlayer = $AudioStreamPlayer  @onready var boost\_timer = $BoostTimer  @onready var speed\_particles = $SpeedParticles  @onready var area\_3d = $Frog/Area3D  @onready var death\_timer = $DeathTimer  @onready var death\_sound = $DeathSound  @onready var collision\_shape\_3d = $CollisionShape3D  @onready var sinkSound = $AudioStreamPlayer3D  @onready var splashParticles = $splashParticles  @onready var colisionCast = $Frog/RayCast3D  @onready var winCam = $Marker3D/winCam  @onready var winRotation = $Marker3D  @onready var playerCam = $Camera3D  @onready var burnSound = $BurnSound  @export var tileDistance = 2.5 #Discrete distance for each movement  @export var moveDistance = 0  var SPEED = 10  var direction = Vector3()  var moving : bool = false #used to determine if a player is moving  var play : bool = false  var startPosition = Vector3()  var dead : bool = false  var lili : int = 0  var liliSpeed = Vector3.ZERO  var overlapping\_lilipads = {}  var rot = -50  func \_ready():  colisionCast.enabled = true  func boost():  SPEED = 20  tileDistance = 5  speed\_particles.emitting = true  boost\_timer.start()  @warning\_ignore("unused\_parameter")  func \_process(delta):  if rot > -65:  rot -= delta \* 5  playerCam.rotation\_degrees.x = rot    winRotation.rotation.y += 0.5 \* delta  if not moving:  var input\_vector = getAction()  if input\_vector != Vector3.ZERO:  direction = input\_vector.normalized() # Convert Vector2 to Vector3  moving = true  moveDistance = 0  startPosition = position  rotateChar(direction)  audioPlayer.play()    play = frog.playing #update the animation states  if moving and not dead:  check\_and\_sink()    if dead:  move\_and\_slide()  else:  velocity = direction \* SPEED  if lili:  var colliders = area\_3d.get\_overlapping\_areas()  for colider in colliders:  if colider.get\_parent().is\_in\_group("Lili"):  liliSpeed = colider.get\_parent().velocity  velocity += liliSpeed  move\_and\_slide()  break    if moving:  play = true  moveDistance = (position - startPosition).length() # Accumulate the distance moved  #print(moveDistance)  move\_and\_slide()    if moveDistance >= tileDistance:  moving = false # Stop moving after reaching the tile distance  direction = Vector3.ZERO    if colisionCast.is\_colliding():  var collider = colisionCast.get\_collider()  if collider and collider.is\_in\_group("Wall"):  moving = false    if play:  anim.play("Armature\_003Action")  else:  anim.stop()  func getAction() -> Vector3:  if Input.is\_action\_just\_pressed("Left"):  return Vector3(-1, 0, 0)  elif Input.is\_action\_just\_pressed("Right"):  return Vector3(1, 0, 0)  elif Input.is\_action\_just\_pressed("Up"):  return Vector3(0, 0, -1)  elif Input.is\_action\_just\_pressed("Down"):  return Vector3(0, 0, 1)  return Vector3.ZERO  func rotateChar(dir: Vector3):  if dir != Vector3.ZERO:  var targetPos = frog.global\_transform.origin + direction  frog.look\_at(targetPos, Vector3.UP)  frog.rotation = Vector3(0,frog.rotation.y,0)  func \_on\_boost\_timer\_timeout():  speed\_particles.emitting = false  SPEED = 10  tileDistance = 3  func die():  death\_sound.play()  death\_timer.start()  func sink():  splashParticles.emitting = true  sinkSound.play()  death\_timer.start()  func burn():  frog.rotation = Vector3(0,-frog.rotation.y,0)  burnSound.play()  death\_timer.start()  func check\_and\_sink():  # Check if frog should sink (not on a lilipad and not dead)  if lili <= 0: # Check if not on a lilipad  var colliders = area\_3d.get\_overlapping\_areas()  for collider in colliders:  if collider.get\_parent().is\_in\_group("River"):  dead = true  velocity = Vector3(0, -1, 0)  sink()  break # No need to check further  func \_on\_area\_3d\_area\_shape\_entered(area\_rid, area, area\_shape\_index, local\_shape\_index):  var colliders = area\_3d.get\_overlapping\_areas()  var parent = area.get\_parent()  if parent.is\_in\_group("Lili"):  if overlapping\_lilipads.has(parent) == false:  overlapping\_lilipads[parent] = true  lili += 1  for collider in colliders:  if collider.is\_in\_group("Win"):  collider.winCon()  winCam.current = true  death\_timer.start()    if collider.is\_in\_group("Rock"):  dead = true  velocity = Vector3(0, -1, 0)  sink()  if collider.is\_in\_group("Fire"):  dead = true  velocity = Vector3(0, -0.5, 0)  burn()  if collider.get\_parent().is\_in\_group("Car") and not dead:  dead = true #flag to stop from bouncing back and forth  var deathVelocity = collider.get\_parent().velocity  velocity = deathVelocity  area\_3d.get\_child(0).set("disabled", true)  area\_3d.monitoring = false  area\_3d.monitorable = false  collision\_shape\_3d.disabled = true  frog.rotation = Vector3(0,frog.rotation.z,0)  die()  func \_on\_death\_timer\_timeout():  get\_tree().reload\_current\_scene()  func \_on\_area\_3d\_area\_shape\_exited(area\_rid, area, area\_shape\_index, local\_shape\_index):  if area:  var parent = area.get\_parent()  if parent.is\_in\_group("Lili"):  if overlapping\_lilipads.has(parent):  overlapping\_lilipads.erase(parent)  lili = max(lili - 1, 0)  func \_on\_splash\_particles\_finished():  splashParticles.emitting = false  velocity = Vector3.ZERO  frog.visable = false |

“frog.gd”

|  |
| --- |
| extends Node3D  var playing : bool = false  func \_on\_animation\_player\_animation\_finished(anim\_name):  playing = false |

“lamp\_post.gd”

|  |
| --- |
| extends Node3D  @onready var marker\_3d = $Marker3D  func \_on\_timer\_timeout():  marker\_3d.rotation.y = randi\_range(0,360) |

“LiliSpawner.gd”

|  |
| --- |
| extends Node3D  @onready var collision\_shape\_3d = $Area3D  @onready var timer = $Timer  var liliPad = preload("res://Scenes/lilli\_pad.tscn") #  var rng = RandomNumberGenerator.new()  func \_ready():  timer.start()  spawn()  func spawn():  var check = collision\_shape\_3d.get\_overlapping\_areas()  var floaters = [liliPad] #array of objects that can spawn  var floater = floaters[randi()%floaters.size()]  var time : float = rng.randi\_range(10, 50)/10  timer.wait\_time = time  var lilis = 0  for checks in check:  if checks.get\_parent().is\_in\_group("Lili"):  lilis += 1  if lilis == 0:  var chance = rng.randi\_range(1, 9)  if chance%3:  var spawnedFloater = floater.instantiate()  add\_child(spawnedFloater)  func \_on\_timer\_timeout():  spawn() |

“lili\_pad.gd”

|  |
| --- |
| extends CharacterBody3D  var SPEED = 2.5  var direction = Vector3(0,0,-1)  func \_process(delta):  var face = global\_transform.basis.x.normalized() \* -1  velocity = face \* SPEED  move\_and\_slide() |

“rock.gd”

|  |
| --- |
| extends Area3D  const ROCK1 = preload("res://Assets/Rocks/BigRock1.obj")  const ROCK2 = preload("res://Assets/Rocks/BigRock2.obj")  const ROCK3 = preload("res://Assets/Rocks/BigRock3.obj")  const ROCK4 = preload("res://Assets/Rocks/BigRock4.obj")  @onready var meshShape = $MeshInstance3D  func \_ready():  var rocks = [ROCK1, ROCK2, ROCK3, ROCK4]  var rock = rocks[randi()%rocks.size()]  meshShape.mesh = rock |

“SpeedBoost.gd”

|  |
| --- |
| extends Node3D  @onready var collision\_shape\_3d = $Area  @onready var sprite\_3d = $Sprite3D  signal speedPower  func \_process(delta):  var camera = get\_viewport().get\_camera\_3d()  if camera:  var camera\_y = camera.global\_transform.basis.get\_euler().x  sprite\_3d.rotation = Vector3(camera\_y,0,0)  func \_on\_area\_body\_entered(body):  var target = collision\_shape\_3d.get\_overlapping\_bodies()  for e in target:  if e.is\_in\_group("Player"):  e.boost()  queue\_free() |

“win\_spot.gd”

|  |
| --- |
| extends Area3D  @onready var meshInst = $MeshInstance3D  @onready var winSound = $AudioStreamPlayer  @onready var winSparkles = $GPUParticles3D  var up : bool = false  var length = 0  func winCon():  winSparkles.emitting = true  meshInst.visible = !meshInst.visible  winSound.play()  func \_process(delta):  meshInst.rotation.y += 5 \* delta  var dist = 2 \* delta    if up:  meshInst.global\_position.y += dist  length += dist  else:  meshInst.global\_position.y -= dist  length -= dist    if length >= 3:  up = false  elif length <= 1:  up = true  func \_on\_audio\_stream\_player\_finished():  queue\_free() |