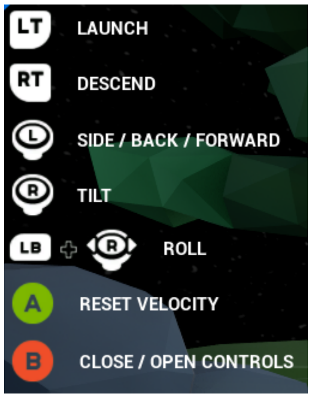
Final Game - Prototype

Game Title: *Rock*

By: *Daniel Bezugliy*

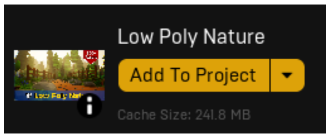
Accessibility / Approachability

- The game has quite a simple premise. The player role-plays as a rock that has gained powers, allowing it to fly and explore the vast outer space. People know what rocks are, there is a deep human fascination with space, and then there are people who wish they were just a rock having no thoughts, so I feel as though many people can pick up and play the game. This simplicity of the game's features and premise shape the game to be approachable, but there were two aspects that hinder that:
  - **Controls and Movement**: I'll talk about movement more in the "Mechanics" section, but the main takeaway is that the movement can be seen as a bit much for some players. In order to counteract this, I wanted to somewhat ease the player into the controls but also still create a surprise with what is possible. *Specifically, I wanted the player to have the feeling of "Oh cool, I can launch myself into space as a rock" and then realize it was just the tip of the iceberg.* Implementing a controls UI mixed with a tutorial section helped make this feeling happen. The tutorial eases the player into the controls, shows how to access the control menu in case clarification is needed, and even shows the danger of being hit by other objects (Further discussed in Design section). I now no longer needed to stand behind the shoulder of each player telling them how to move properly.



LT	LAUNCH
RT	DESCEND
L	SIDE / BACK / FORWARD
R	TILT
LB + R	ROLL
A	RESET VELOCITY
B	CLOSE / OPEN CONTROLS

    - Additionally, in the case that things get too out of hand and to help with better control of the player (since it might be hard to see if the rock is even moving when compared to empty space), a button was added to reset all velocity.
  - **Unreal Engine**: Since the game is implemented using the Unreal Engine, I immediately had concerns about people (mainly myself) even being able to launch my game. In order to counteract the hardware limitations needed for an open-world Unreal Engine space game, low-poly assets and meshes were used for almost everything. This of course wasn't an end all solution, but I believe it made some positive difference as at least my old laptop could run the game during class.



- **Additional Accessibility Issues Considerations**: Hand Dexterity was a major concern when developing the game due to my prior experience with the game Outer Wilds, but it was also that prior experience that provided a solution. When playing Outer Wilds for the first time, I ignored the "Best Experienced with a GamePad" text, paid the price, and learned how difficult (and uncomfortable) it was to move around in zero gravity space with just keyboard/mouse. Due to this, I designed my game with the ability to use a controller and even have my own "Best Experienced with a GamePad" text.

- **Approachable for Non-Intended Audience**: For those who don't care to role-play as a rock or mess around with just movement in space, there were two additional aspects implemented in the game to keep certain players engaged:
    - *Achievement Hunter*: There are small carrot collectables scattered across different planets that players can… collect. These carrots grow the player's size, and if all the carrots are collected, an achievement is awarded. *Yipee*.
    - *Destructionist*: For those who just want chaos, collision with the environment was implemented (only after a bunch of headaches). If the player crashes into the ground, the rock explodes. This also acts as an end/reset to the game.

Mechanics

- The game "Rock" is an exploration game, so ***movement*** was the main focus:
    - With a newly formed identity as an asteroid, the player has different directional thrusts, rotations, and rolls that they have access to. In order to get a better viewpoint, players will also need to know how to orient themselves. Velocity is conserved between actions and there is no air resistance or drag to slow the player down, meaning extra caution is needed.
    - If the player feels confident in their movement, they can test it through ***collectables***:
        - As mentioned previously, players can collect carrots scattered across space in order to grow the player's size. The bigger the player grows, the harder it might become to avoid crashing when going after other collectables.

    - If they player is careless with movement, they might **crash**:
        - If the player is careless with their movement and surroundings, they have the chance to hit something and explode. This introduces the need for caution in the game. Initially there was a minimum speed that the player had to be traveling to cause an explosion, but I thought it was way funnier to hit something very gently and violently explode.

- With movement at its core, there is a lot to mess around with in the game to hopefully keep the player engaged role-playing as a rock. The mechanics of collection and crashing also blend smoothly together to create a risk/reward dynamic.

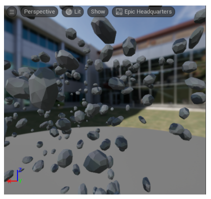
- *NOTE: I was hesitant to add the collection mechanic since it might not make sense with the core loop of launching into space and crashing back down, but a lot of people showed interest in the katamari approach and growing their rock size, so it ended up being implemented. I think it was a good decision since as mentioned previously, it is a good way to test/challenge the player's control of movement.*
- *NOTE: Like collectibles, the crashing feature was added towards the end development of the game. Unlike the collection mechanic, the destruction/collision mechanic was always meant to be implemented but took a while due to the difficulty of implementation. The properties of the player as a pawn class and the fact that objects don't actually affect the velocity of the player due to how movement was implemented were the main culprits. If I were making this game all over again, I would implement movement differently, but due to limited time, I couldn't start from scratch. The end result isn't perfect (and there are probably problems with it), but I eventually found some workarounds with Unreal's chaos engine.*

Design & Rules

- *You're a rock in space and you can do whatever you want! See that planet in the distance? You can go to it! See the tree below you? You can crash into it! Don't want to do anything? Don't! You are just a rock after all.* This is the message the game aims to send, and although it sounds freeing as I believe exploration games should feel, there are obviously restrictions such as:
    - There are limited controls presented to the player that must be abided by.
    - If the rock crashes, the player must either reset or quit.
    - The tutorial section is non-skippable and must be sat through.
    - You can only collect each carrot once.
- Since the game is aimed to be an open-world, exploration game, there is only one level: Outer Space. Although there is only "one level", different aspects of it were designed with a clear purpose.

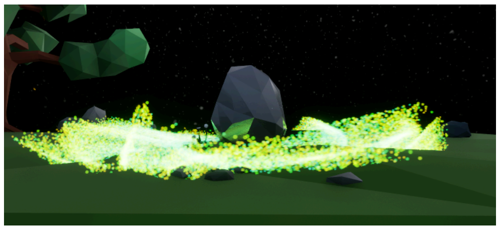
Outer Space Environment:

- To feel like the player is actually in space, there is a space skybox, no boundaries, planets, zero gravity, and… no shadows? Since there isn't a light source, there can't be shadows (this also means everything should be pitch black, but let's pretend the rock has an all-seeing eye for the sake of playability).
- One aspect that is untold in the story of the game is that the player's motivation as a rock to go and fly up is that it got jealous of asteroids. So, asteroids are implemented and can be seen flying around in space at random.



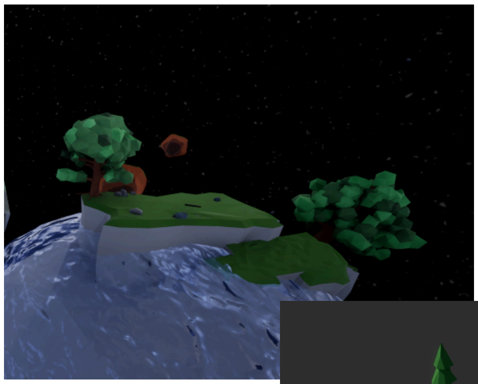
Tutorial

- A series of text from “god” is showcased to give an introduction to the game. An aspect that I thought turned out really cool is that the text is actually placed in the 3D environment, so when the player flys up, they leave the text behind.
- Particles are implemented to give the feeling that the player gains “power”, thus causing it to fly up. Note: If you look closely, there are also particles of falling leaves from the tree to make the environment a bit more interesting.

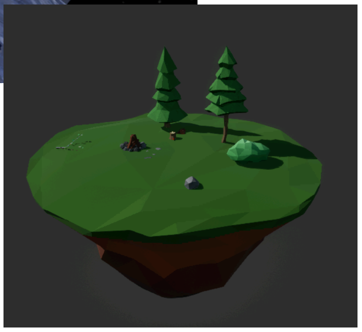


- An island is spawned when the player launches up and is set up in a way to be on a collision course with the player. This was intended so that the player understands that ‘with freedom comes risks’. Only after this event are all the controls given to the player. Also, I aimed the collision in such a way that when the player goes spinning out of control, they have a chance of seeing the other planet (introducing the idea that there’s more to explore).
- Overall, the tutorial was implemented in a way to give an introduction to the game, ease the player into the controls, and show the dangers of being hit.

Home Planet



- Simple, stationary, and detailed. This planet is where the player starts the game. The trees and ground are collidable but the water isn’t. Apart from the spawn island, there is a small island with some structure ruins and a carrot. There are also two floating islands, one with a campfire that moves fast and another far in the distance (each with a carrot).
- NOTE: Carrots are placed in areas either to reward exploration and/or reward movement. For example, if the player can match the rotation and speed of the floating islands, they can collect the carrots placed there.



Rotating Planet

- Similar to the home planet, but this time the planet is rotating slightly. Again, in order to collect the carrot placed there, they must match the rotation.

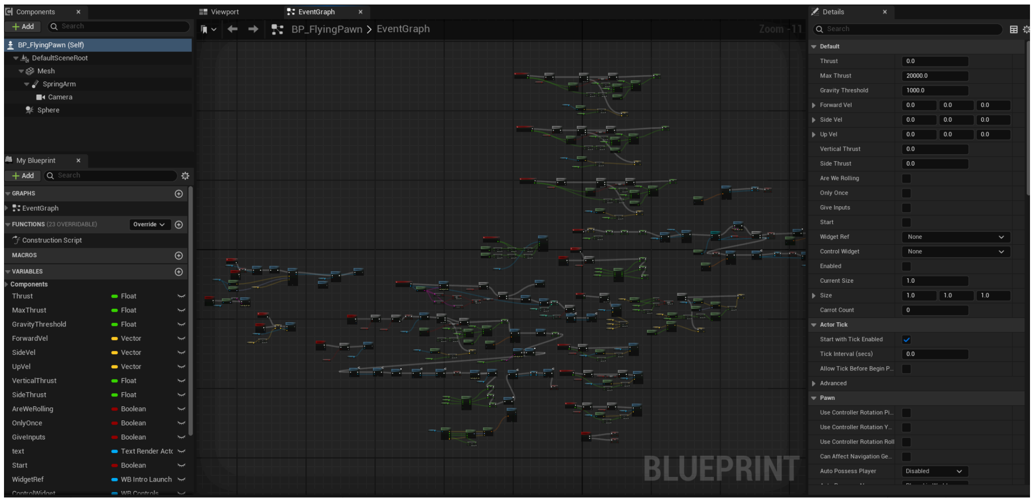
Chaotic Planet

- This planet is just barren and chaotic. Good luck.

NOTE: The entire layout of the game was designed and placed by hand, but I did not make the assets. For the UI, I used free assets from Kenney’s (https://kenney.nl), pre-owned Fab Lab materials for the low-poly meshes, and Unreal’s built-in niagara particle system for all the particles (tutorial and asteroids).

Concept

- By using the materials listed above and implementing the mechanics/design as mentioned, I believe that I was able to execute the vision of ‘being a rock’. The tutorial helps the player get started, there is an end to the game by crashing, the outer space environment turned out great, and there are more details that helped make the game feel like a game. More can be added and aspects can be changed, but I’m still happy with the end product.



Components:
- BP_FlyingPawn (Self)
  - DefaultSceneRoot
    - Mesh
      - SpringArm
        - Camera
    - Sphere

My Blueprint:
- GRAPHS
  - EventGraph
- FUNCTIONS (23 OVERRIDABLE) — Override
  - Construction Script
- MACROS
- VARIABLES

Variable	Type
Components	
Thrust	Float
MaxThrust	Float
GravityThreshold	Float
ForwardVel	Vector
SideVel	Vector
UpVel	Vector
VerticalThrust	Float
SideThrust	Float
AreWeRolling	Boolean
OnlyOnce	Boolean
GiveInputs	Boolean
text	Text Render Actc
Start	Boolean
WidgetRef	WB Intro Launch
ControlWidget	WB Controls

Details:

Default			
Thrust	0.0		
Max Thrust	20000.0		
Gravity Threshold	1000.0		
Forward Vel	0.0	0.0	0.0
Side Vel	0.0	0.0	0.0
Up Vel	0.0	0.0	0.0
Vertical Thrust	0.0		
Side Thrust	0.0		
Are We Rolling	☐		
Only Once	☐		
Give Inputs	☐		
Start	☐		
Widget Ref	None		
Control Widget	None		
Enabled	☐		
Current Size	1.0		
Size	1.0	1.0	1.0
Carrot Count	0		
Actor Tick			
Start with Tick Enabled	✓		
Tick Interval (secs)	0.0		
Allow Tick Before Begin P...	☐		
Advanced			
Pawn			
Use Controller Rotation Pi...	☐		
Use Controller Rotation Y...	☐		
Use Controller Rotation Roll	☐		
Can Affect Navigation Ge...	☐		
Auto Possess Player	Disabled		

This is the blueprints of the character class, showcasing how everything works together to make the game function.

CRITIQUES, PROBLEMS, BUGS

There are a lot of problems with the game, but I have limited time and could not get around to fixing them. If I had more time, here are some of the problems I would address.

- The tutorial section was implemented based on the in-game ticks. This means, every computer that runs the game will run the tutorial differently, which is a big concern because the tutorial includes a perfectly timed collision course with an island. On the PC that I developed my game on, the tutorial runs well, but on my lower-end laptop, I never even see the island. *Simple yet not simple change*: do not use the tick action event.
- I hate the movement that I initially implemented, but I was too far into the project that I couldn’t switch everything around. Because of this movement, collisions are all messed up, which makes the destruction mechanic a pain to get working. The destruction still doesn't work how I would like (If a player goes really fast into the ground, they won’t even see the explosion). *Change*: Use built-in velocity and impulse, not ‘lerp to’ function.
- *Change*: More content and more detail is needed to make the world feel alive.
- ~~The tutorial starts playing on the title screen… This doesn’t affect the actual game, but it spoils content and looks bad.~~ ~~*Change*: Remove player spawn at the title map (FIXED).~~