Can You Build Psyche? - Software Design Documentation

NASA Psyche Web-Based Game 18C

Ava Caiola, Austin Holland, Erik Johnson, Brett Koppelman, Nan Li, Richard Marshall

Cassie Bowman

Functional Description

This project is all about the creation of a 2D game that allows players to explore one of the theories about how the Psyche asteroid formed. The game will consist of several user interface elements and game objects. The user interface elements include a title screen, main menu, instructions, tutorial, credits, pause menu, scoring page, and input fields for setting the parameters of the asteroid to collide with Psyche. The game objects include three types of particles: metal, mantle, and crust.

The gameplay revolves around having a player set the parameters of an asteroid. These parameters include mass, velocity, angle of launch, the material the asteroid is made of, and the location the asteroid launches from. Based on the inputs the asteroid that is launched may or may not knock particles off of Psyche. The player can continue to launch asteroids at Psyche until they believe they have reached the point where the Psyche in-game represents Psyche in real-life. A score will be given to the player and the game will explain where they could have done better in trying to create Psyche as it is today.

As for the menus in the game, they will function like how you would expect any menu in a game to function. The title screen states the title of the game and provides a button to go to the main menu. The main menu has a list of options which include: Play, Instructions, Tutorial, Credits, and Exit. Clicking any of the buttons will take the player to that menu or into the playable game. The instructions page will contain a quick explanation of how to play the game. The tutorial page will contain a more detailed explanation of how to play the game and include pictures to help show how to use the UI in the game to set the parameters of the asteroid. The credits page will list all the team members who worked on the game. The exit button will exit out of the main menu and return the player to the title screen.

The pause menu can be accessed by pressing a pause button located in the UI of the game. Clicking the button will pull up the pause menu and cause the game to pause also. Within the menu, four options will be given to the player. These include resume, restart, settings, and exit. The resume button unpauses the game and causes the pause menu to disappear. The restart button resets the game and allows the user to begin launching asteroids again. The settings button allows a soundbar to show up. The player can use the soundbar to adjust how loud the game is. The exit button returns the player to the main menu of the game.

Some additional design decisions in the game include how particles are attracted to each other, how the game knows which particles were knocked off of Psyche, and how the game handles timing all the systems working in the game.

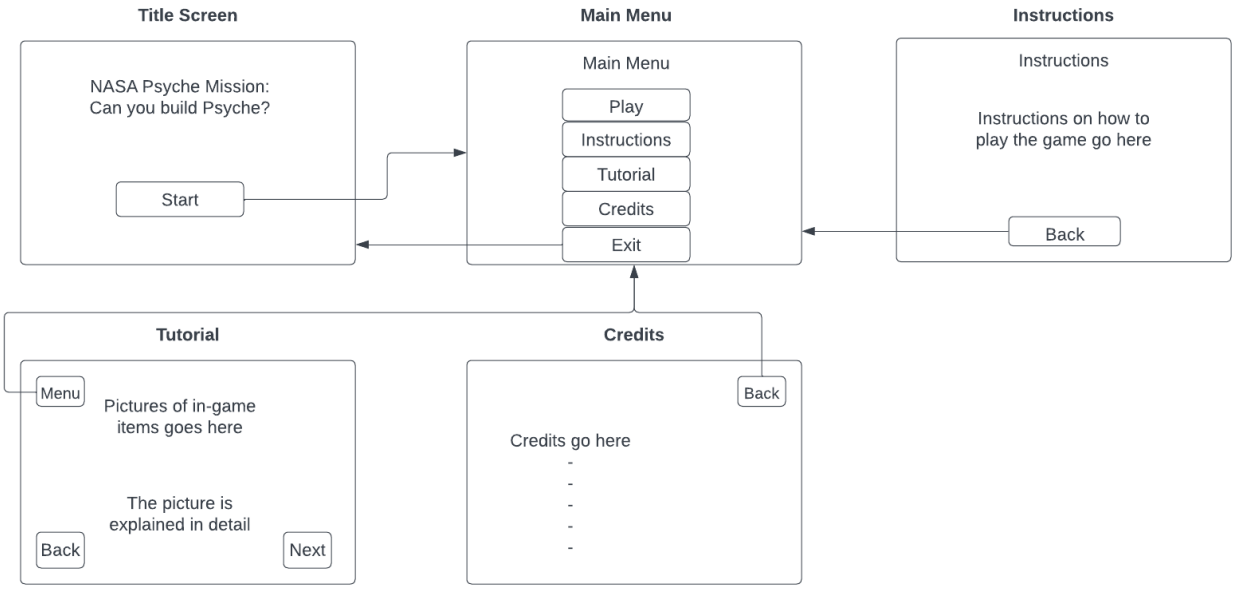
Particle attraction is handled by a script that applies gravity to the different particles. The script looks at the mass of each particle and uses it to calculate the different amounts of force each particle feels to other particles. This force is applied to the particles using Unity's existing components resulting in the particles being attracted to each other. The script is applied to every particle which means each particle has an attraction to other particles. This script is to be disabled on the asteroid to be launched to prevent Psyche from moving toward the unlaunched asteroid. The script is enabled upon launching the asteroid.

The game keeps track of what particles are a part of Psyche by using a script to propagate a boolean that is contained within the script. The boolean is set to true if the particle is in contact with a particle that already has the boolean as true. Using this boolean particles that were knocked off of Psyche by a collision can be removed from the current list of Psyche particles. The script is by default enabled on all the particles with the boolean set to false, except for the central Psyche particle. This particle has the script by default disabled because it will always have its boolean set to true. The script is enabled from within the timer script after about 10 seconds have elapsed since the player launched the asteroid.

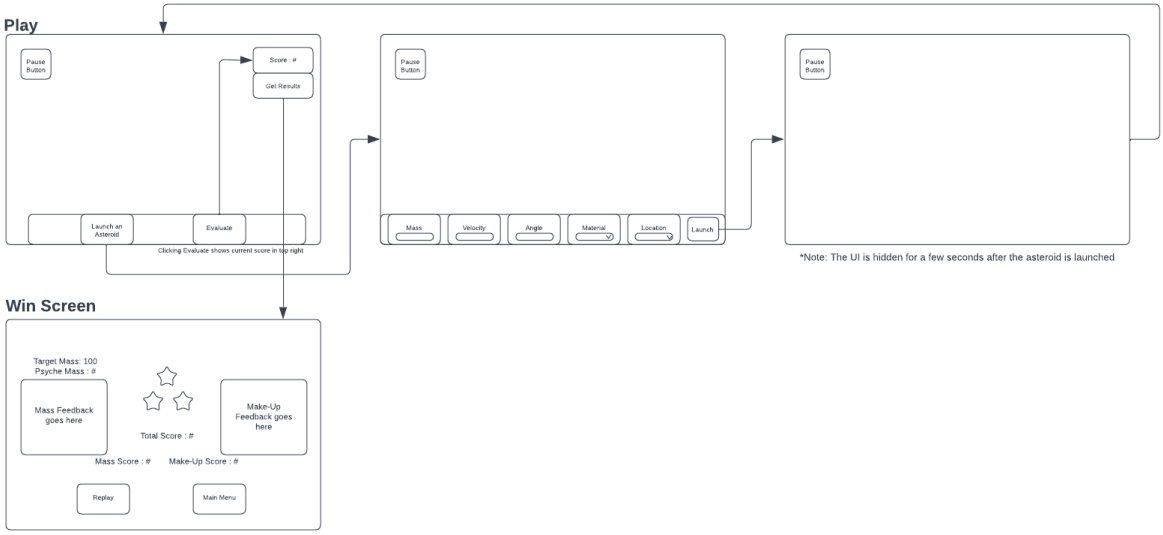
Creating a script to act as a timer was critical for the game. The script controls the order in which everything happens. After the asteroid is launched the UI is hidden from the player. This allows the player to fully see the game and have a better experience. After 1 second the script on the middle Psyche particle is enabled. After 5 additional seconds, a ParticleCounter.cs script is called to go through and count and add the particles that are part of Psyche to a list of particles. The script then sets all the velocities of the particles to zero and resets Psyche to a default point in the game. The asteroid that was launched has its tags changed to make scripts stop recognizing it as the asteroid to be launched and to see it as an average particle in the game. The UI is then reenabled and the player can continue playing the game.

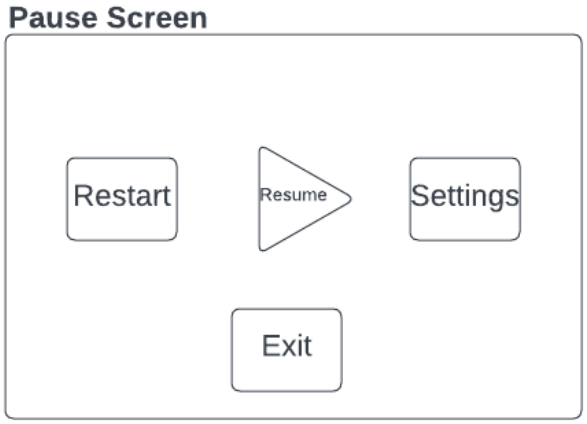
User Interface

Menus:



In-Game:





Milestones:

1. Complete the menus for the game.
2. Implement the 1st version of the particle system.
3. Create the in-game UI
4. Implement particle tracking and make the system know what particles were knocked off in a collision.
5. Implement scene resetting.
6. Implement scoring functionality.
7. Start bug testing and write unit tests.
8. Release the game into beta.
9. Submit the final version of the game.

Timeline:

In-Depth Script Descriptions:

Script	Public Variables and Methods	Description

UserInputToAsteroid.cs	boolean ClickedButton;
	int massInput;
	int velocityInput;
	int angleInput;
	string materialInput;
	TMP_InputField massInputField;
	TMP_InputField velocityInputField;
	TMP_InputField angleInputField;
	GameObject otherAsteroid
	
	void StoreParameters();
	void HandleAngleInput();
	void DeleteProjecttedLine();
	void HandleMaterialInput(int val);
	void HandleLocatioInput(int val);
	void setOtherAsteroid();
	void Awake();
		This script is ran when the “Launch” button is pressed in-game. The StoreParameters() method starts by reading the mass the user inputted and checks if the entered value is within acceptable ranges. It does the same for velocity and angle inputs. It then takes the value from the material dropdown and applies that property to the otherAsteroid. If any of the inputs are invalid the script prevents ClickedButton from being true and doesn’t launch the asteroid. If no issues are found with the inputs the parameters are applied to the asteroid.
		
		The HandleAngleInput() is called whenever the user enters an angle and presses enter or clicks off of the angle input field. This methods is responsible for drawing the projected launch line in-game
		
		DeleteProjectedLine() is ran when “Launch” is pressed and when the user changes the angle in the input box(so the original line doesn’t remain). It destroys the ProjectionLine GameObject.
		
		The HandleMaterialInput(int val) method is used to simply set a private int called materialSeleted to 1, 2, or 3 depending on the material the user selected. It is called whenever the user selects a material.
		
		The HandleLocationInput(int val) method is also called whenever the user selects a different location. Based on the selection the otherAsteroid’s location is moved to that position. The locations are determined by looking at Psyche’s center particle location and adding a

		Vector3 to that location. void setOtherAsteroid() is called after a scene is reset. It changes the script's reference to the original otherAsteroid and sets it to the new one.
AfterLaunchTimer.cs	List<GameObject> particles; GameObject parentHUD; void Start(); IEnumerator Countdown();	This script controls all the events that happen after the “Launch” button is pressed that don’t evolve reading user inputs. It also is respnsibe for resetting the scene within the game. This script is attached to the otherAsterod GameObject and is disabled initially. The script is only enabled from the StoreParameters() in UserInputToAsteroid.cs after all userInputs have been applied to the otherAsteroid and it has been launched. This is the order in which events occur within the script. - Wait 5 seconds - Enable the PsycheParticleCounter script on the MiddleAsteroidParticle - Wait 5 more seconds to allow the PsycheParticleCounter to propagate a boolean between all particles that remain part of Psyche after a collision - Call the countWhatParticlesArePartOfPsyche() in the ParticleCounter GameObject. - Set all velocities of the particles to zero - Move Psyche to a default position within the scene - Set all velocities to zero again to prevent abnormal behavior. - Disable the PsycheParticleCounter script on the MiddleAsteroidParticle GameObject.

		- Parse through all the particles that are part of Psyche and set the partOfPsyche boolean contained within their PsycheParticleCounter.cs script to false - Change the tags of the previous otherAsteroid - Wait 1 additional second. - Generate a new otherAsteroid to the left of Psyche - Move the new otherAsteroid into the GameObject parent called “Other Asterod” - Call the setOtherAsteroid() method in UserInputToAsteroid.cs - Reeable the “HUD Parent” UI

Atteactor.cs	static List<Attractor> Attractors;
	RigidBody2D rb;
	
	void FixedUpdate();
	void OnEnable();
	void OnDiable();
	void Attract();
		This script has been attached to all the particles within the game. It is responsible for creating the gravity between the particles.
		
		The FixedUpdate() method is used to have the forces be applied 50 times per second. Within the method is parses through the list of Attrators and calls the Attract().
		
		OnEnable() adds particles to the Attractors list when the particle is 1st enabled
		
		OnDisable() removes particles from the list when they are disabled.
		
		Attract() is used to calculate the force two particles feel between each other and it applies that force to the rigidbody2D of the particle.

ParticleCounter.cs	int metal_particle_count;
	int mantle_particle_count;
	int crust_particle_count;
	
	void Start();
	void countWhatParticlesArePartOfPsyche();
	void RemoveParticleCount(GameObject particle);
		This script counts the number of particles on initial game start and after the game resets.
		
		The Start() counts the number of particles in the system when the

	int getMetal_Particle_Count();
	int getMantle_Particle_Count();
	int getCrust_Particle_Count();
		game 1st starts.
		
		The countWhatParticlesArePartOfPsyche() uses a boolean stored within the PsycheParticleCounter.cs script that is attached to every particle to add ot remove from the count determined by the Start(). If particles were knocked off of Psyche those particles are deleted.
		
		The RemoveParticleCount(GameObject particle) looks at the tag of the GameObect and subtracts 1 from that particle type's count.

PsycheParticleCounter.cs	boolean partOfPsyche;
	
	void Start();
	void OnEnable();
	void OnCollisionStay2D(Collision2D collision);
		This script contains the boolean partOfPsyche. This is critical for determining what particles are part of Psyche after a collison has occurred.
		
		The Start() method makes the partOfPsyche boolean true for the middleAsteroidParticle of Psyche.
		
		The OnEnable() method also makes the partOfPsyche boolean true for the middleAsteroidParticl.
		
		The OnCollisionStay2D(Collision2D collision) method propagates the partOfPscyhe boolean between all the particles of Psyche. Initially, the script is disabled on the MiddleAsteroidParticle but it is enabled from within the AfterLaunchTimer.cs.

AngleProjection.cs	float distance; int angle; void DrawProjection();	This script is used to draw the aim assist line from the otherAsteroid.
UI_Hiding.cs	void HideParameterUI();	This method is called when the "Launch" button is pressed. It looks to see if the buttonClicked boolean in the UserInputToAsteroid is true

		before disabling the UI.

TestAsteroidSpeed.cs	float speed;
	void Start();
	void OnCollisionEnter2D(Collision2D collision);
		This script originally was used to launch the otherAsteroid when we were building the particle system. It is now used to disable the Attractor.cs script on the otherAsteroid until it hits Psyche. Once it hits Psyche the Attractor.cs script is reenabled and the particle begins to have gravity to the other particles.

Game_UI_InitialSettigs.cs	GameObject UserParameters_HUD;
	void Start();
	void ActivateUserParameters_HUD();
		This script is used to disable and enable the UserParameters_HUD.

TutorialButton.cs	Button Back;
	Button Next;
	List<GameObject> Pages;
	void Start();
	void PrevPage();
	void NextPage();
		This script controls which page is active within the tutorial. As the user presses the “Next” and “Back” buttons it changes the scene to the correct page.

TextScroll.cs	void Start(); IEnumerator AnimateText();	This script is used by the credits page to make the text appear as if it is being typed out.

SceneLoader.cs	void LoadScene(string sceneName);
	void Pause();
	void Resume();
		This script is used throughout the game to load new scenes.

Sound_slider.cs	AudioMixer audioMixer; void setVolume(float volume);	This script is used to control the volume level within the pause menu.
scoreStar.cs	GameObject star1; GameObject star2; GameObject star3; TMP_Text scorePoints; TMP_Text massFeedback; TMP_Text currentMass; TMP_Text massScore; TMP_Text makeUpScore; TMP_Text otherParticleFeedback TMP_Text starCongrats void Start();	This script is used to adjust all the text fields within the win page of the game. It reads data from a data holding script and uses the data to determine what text should be displayed to the player.