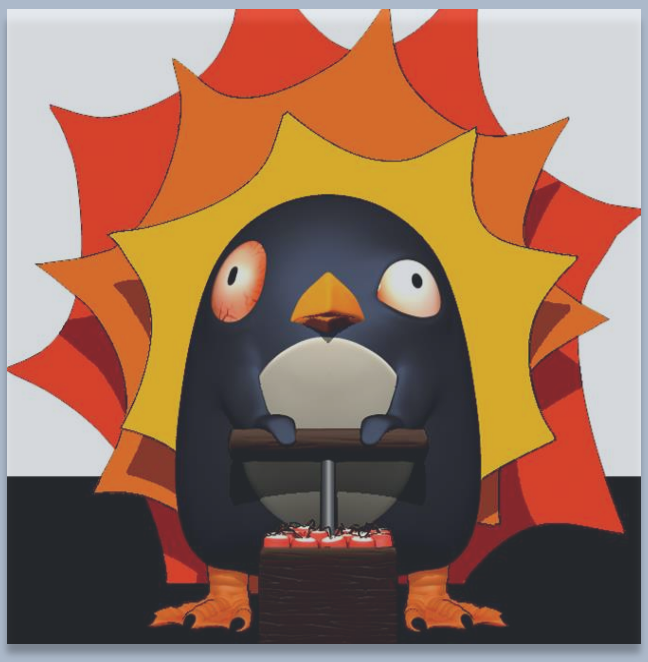
PROOF OF CONCEPT - TECHNICAL DESIGN DOCUMENT

By Len Farag and Thomas Dufresne



Content

1.0 Development Environment

1.1 Game Engine

Unity 2020.3.5f

1.2 IDE

Visual Studio 2019

1.3 Source Control

Git (via GitHub)

- GitKraken (GUI based)
- GitBash (console based)

1.3.1 Source Control Procedures

Template: [ASSET TYPE] [CHANGE TYPE] [SHORT SUMMARY]

Asset Types: CODE, ART, SCENE, AUDIO, PACKAGE, SETTINGS, REFACTOR, RESYNC

Change Types: ADD, REMOVE, FIX, CHANGE

Examples:

- [ART] [ADD] [Add 3 new player models]
- [CODE] [FIX] [Fix camera drift in character controller]

Commit procedures:

- Frequency: Once at start of day, once each time a completed change is made. This means commits should not be committed if existing code is broken because of a push, but rather should only be pushed once a completed and sustainable change is made.
- Summaries should be in present tense (add vs added).
- Each pull should be a rebase to prevent branching.
- Summary should be short, and descriptive. If more description is required, use a comment.
- Scenes should be “checked out”, that is, communicated to the rest of the team that it is being worked on, and should not be changed by any developer while it is being worked on. This will prevent merge conflicts.
- Binary files should be limited to small file sizes, unless crucial.
- If assets such as audio files are being sourced, care should be taken to minimize asset bloat, and only finalized assets should be submitted.

1.4 Third-Party Libraries

- Input System – version 1.0.2
- Universal Render Pipeline – version 10.4.0

- TextMeshPro – version 3.0.4
- FMOD for Unity – version 2.02.03

1.5 Other Software

- Microsoft Office Suite
- Draw.io
- Maya 2020
- ZBrush 2021
- Substance Painter
- Substance Designer
- Photoshop 2020
- Clip Studio Paint
- FMOD Studio

2.0 Game Overview

2.1 Genre

Manager

2.2 Perspective

Orthographic.

2.3 Platform

- PC
- Xbox One

2.4 Technical Goals

- 3D graphics.
- 60 fps locked on Xbox One and PC.
- Open ended systems to allow for expansion.
- Dynamic controller support allows for differing numbers of players.
- Intuitive and simple interaction system.
- Dynamic music system that reacts to gameplay.

2.5 Game Objects and Logic

- Playable characters – prefabbed objects that are instantiated when the game starts. Depending on how many input devices are registered, and how many have confirmed participation, that number of players will be instantiated with the selected player model.
- Controller Manager – a system that will manage connected controllers and determine how to handle controller connections and disconnections in the different stages of the game.
- Character selection – a scene where controllers can be connected and can join the game. In this screen also, each player with a joined controller can select a character that they will play as.
- Ship selection – a scene where the ship (map) is chosen.

- Level selection – a scene where the level for the game is selected.
- Station – a complex object using interactors to allow players to affect its state. Stations include the reactor, engines, turrets, and scanners. Hull holes are typically referred to as stations but are a unique interactable.
- Interactables – an object that can be interacted with by players (via interactors). Mostly used by stations, and include switches, fuel tanks, repair stations, and others.

3.0 Controls

Keyboard controls exist, but are only active while in the editor, and are disabled in builds.

3.1 Xbox One Controller

- A Button
  - Interact with station.
  - Repair hole.
  - Pickup grabbable item.
- B Button
  - Return.
  - Drop grabbable item.
- Left Stick
  - Move character around.
  - Move character selector.
- D-Pad
  - Move character selector.

3.2 Keyboard

- E
  - Interact with station.
  - Repair hole.
  - Pickup grabbable item.
- Q
  - Return.
  - Drop grabbable item.
- WASD
  - Move character around.
  - Move character selector.
- Arrow keys
  - Move character selector.

4.0 Mechanics

4.1 Oxygen

- The ship's oxygen level represents its health, and players lose if it reaches 0.
- Hull breaches deplete the total oxygen level. Each hole will drain a set amount per second, as set by designers.

- The rates are additive. However, there is a cap to max oxygen leakage to prevent insta-death scenarios.
- Oxygen is generated by the reactor while it's turned on.

4.2 Reactor

- The reactor is a station with 4 interactables: a repairable, a fuel depo, and 2 switches.
- Fuel is generated while the reactor is turned on and will become available at the fuel depo when it's ready.
- While the reactor is turned on, the ships' oxygen level will slowly increase.
- If the reactor is damaged, it will be turned off and needs to be repaired before it can be turned on again.
- To turn the reactor on or off, two players must interact with a pair of switches in a short time window.

4.3 Turret

- Turrets are stations with 3 interactables: a repairable, a fuel tank, and an activator.
- The fuel tank can be filled by interacting with it while holding a fuel grabbable. On completing this interaction, the fuel grabbable will be consumed.
- While the turret is fueled and not damaged, the activator can be used to use the turret. While a player is using the turret, the character controls are disabled, with the left stick used to aim the turret and the A button used to fire. The B button is used to exit the turret.
- Projectiles fired from the turret can hit asteroids and destroy them, preventing damage to the ship.
- If a turret is damaged, it can't be activated until it has been repaired.

4.4 Asteroids

- When an asteroid hit the ship, the hull section that is impacted suffers a hull breach, and nearby stations have the possibility of becoming damaged.
- Asteroids are spawned off screen with a UI indicator showing where they will come from.
- Asteroids can be shot down with turrets to prevent damage to the ship.

4.5 Level/Events

- Engines that are turned on will contribute to the ship's speed, moving it along the level. Players win when they reach the end of the level.
- Throughout the level are events that begin when the ship passes its starting position, and end when the pass its ending position, or otherwise trigger it to end.
- Levels are represented by the scanner UI at the top of the screen in the form of a timeline containing the player’s ship and, when the scanner is used, events in the level.

4.6 Engine

- Engines are stations with 2 interactables: a repairable, and a fuel tank.
- The fuel tank can be filled by interacting with it while holding a fuel grabbable. On completing this interaction, the fuel grabbable will be consumed.
- If an engine is damaged, it will be turned off until it is repaired.

- When an engine has fuel and is repaired, it will be turned on.
- While turned on, the ship's speed for the level will be increased, and fuel will be consumed from the fuel tank after a set amount of time has passed.

5.0 Systems

5.1 Controller Manager

- Manages input devices (gamepads), and participation.
- At the character selector, gamepads that are connected to the console will show as a joinable slot.
- If the player presses the confirm button on the controller to join, it will assign a character selector to the player.
- Once the game moves past the character selector and player inputs are locked in, the system will manage controller connections and disconnections. If a controller that is assigned to a player is disconnected, a “Reconnect controller” state will be set.
- If the controller is reconnected, it will allow the player to resume control of their character.
- The manager should take care to only assign 1 player to 1 character controller.

5.2 Character Selector

- In the character selection screen, each player who has joined will be in control of a selector.
- The select can move around a grid of avatars, and each time the character is changed, the model is loaded into a preview section on screen.

5.3 Ship Manager

- Each ship prefab has a ShipManager component on its root object.
- The ship manager handles the ships oxygen level, speed, and taking damage from asteroid impacts and plasma storms.
- A room manager is used to allow for the possibility of a ship composed of multiple rooms, although that has not been the case, and instead a single room manager is used for the whole ship.
- The room manager handles hull hole stations within the room, notifying the ship manager when its total oxygen drain is changed.
- When an asteroid collides with the ship, it calls a function passing the contact position. This function uses each room manager in the ship to find the hole position closest to the contact position, then tells the corresponding room manager to create a hole at that hole position and roll a chance for nearby damage stations to become damaged.

5.4 Interaction

- The interaction system consists of 4 scripts: Interactor, Interactable, Grabbable, and InteractionManager.

- The interaction manager contains lists of all interactors and interactables, but its primary purpose is to aid in managing interaction prompts. Interactors and interactables can pass necessary data to the interaction manager, allowing it to display prompts as desired. This decision was made as a result of uncertainty about how interaction prompts should be displayed, allowing faster iterations.
- Each player has an interactor component, while each type of interactable object will have a component derived from interactable. Grabbable objects will have a grabbable component, which itself derives from interactable.
- The interactor will setup callbacks for the started and canceled events of player input for the A and B buttons on entering the game scene. When the started event begins, it will iterate through all interactables to find one that it is within the interaction radius of, uses the pressed control, and can be used according to a function using derived logic to allow for interactables to only allow for interactions when a case has been met. When the closest valid interactable has been found, it will begin an interaction with it.
- Interactions between interactors and interactables continue until the *EndInteraction* function is called on the interactor. While interacting, the player will be unable to control their character. The interactable being interacted with will be notified of player input events during this time.
- Locking the player into interactions allows the interactable to take control of the player, as is the case with turrets. When an interaction should only consist of a button press, the interactable can immediately end the interaction to unlock the player.
- Interactables have abstract functions for interaction start and end, as well as button down and button up.

5.5 Level Manager

- A level consists of its length and an array of events. Events are represented in the level with a start and end position (floats) and use an enum for the type of event it represents.
- A designer is able to set difficulty related data using a difficulty settings scriptable object.
- The player ships speed is determined by the number of engines that are turned on.
- When the player ship's position passes the starting position of an event, it is activated, and becomes deactivated when it passes the end position, unless the event is ended prematurely by other means.
- Event objects have start, end, and update functions, and are instantiated when the event begins with values set by designers.
- Currently, the 3 events include asteroid fields, plasma storms, and ship attacks. Asteroid fields use the asteroid manager to spawn more asteroids. Plasma storm uses the ship manager to damage stations that are turned on. Ship attack instantiates a ship to follow the player ship and fire missiles at it. Missiles are reskinned asteroids, and players can use turrets to shoot the ship and destroy it before exiting the event area.

- When the player ship's position reaches the level length, the game ends with a win state.
- If the player selects endless in the difficulty select, an alternative game mode is used. Rather than the player ship reaching the end of the level, events are instead constantly generated after the last is completed, with the player ship remaining stationary in the scanner UI and events moving towards it.
- To accomplish the endless mode, levels have the added functionality to add and remove events dynamically. To increase the difficulty over time, the level controller will increment certain difficulty related variables using values in the designer set difficulty settings.

5.6 Asteroid Manager

- The asteroid manager handles setting up asteroid spawn areas, spawning asteroids to either hit or miss the ship, and creating asteroid prompts.
- Asteroids can spawn within trigger volumes that are to the top, left, and right of the camera. These triggers are created on startup using the camera's position and forward direction to determine the screen's center point relative to the player ship's Y plane. This along with the camera's rotation and size is used to create the triggers off screen.
- On startup, a 2D polygon collider on the canvas is also shrunk. This collider is used for placing asteroid prompts so that they don't overlap other HUD elements.
- A public function is used to spawn asteroids. It selects a random trigger volume and gets a random position inside it. Next it determines a direction that will result in an asteroid either hitting or missing the player ship using a raycast. The spawn position and direction are then transformed into canvas space, where a raycast is done into a 2D polygon collider to determine the position for the asteroid prompt.

5.7 Music Manager

- The music manager handles transitioning music between scenes and creating dynamic music that reacts to gameplay.
- A scriptable object is used to store music information, including the FMOD event and which scenes to use it in. A function is used to get the data for a given scene.
- A callback for active scene change is used to determine when a new scene is loaded, and if the music data for the new scene is different to the previous scene, the new music will be played.
- The music manager also allows for events to be called on a beat of the music, with optional delays for both the bar and beat. This can be used to time events with the music.
- The game music will react to the ship's oxygen level and events by setting parameters in the FMOD plugin. The oxygen level will affect layers and instruments, with events causing the music to change to an alternate version.

6.0 Graphics

- The camera will have a set position for each ship and use an orthogonal projection.

- The camera object will consist of two different cameras: one will render only the UI to always have it rendered on top, the second will render the game area (players, asteroids, walkable area of the ship).
- Asteroid Markers: Markers will appear on the edge of the screen with an arrow to show where asteroids are coming from.
- UI above the station will display the control used to interact with the station.
- Hull breaches will have replaced meshes.
- Rotating spot lighting when reactor goes critical.
- Asteroid impact particle effects.

7.0 Artificial Intelligence

- NA

8.0 Physics

- The ship will remain stationary with the background giving the illusion of movement, allowing the ship colliders to be static.
- Grabbable objects will have colliders and rigidbodies set accordingly when picked up and dropped.
- Players are moved by directly setting their velocity to provide consistent movement.

9.0 Game Flow

- The whole game is divided up into scenes.
- These scenes each have a scene controller which oversees managing all interaction while the scene is active. Alternatively, scripts can add a listener to an event in GameManager that is called when the scene is changed.
- All scenes are changed by the GameManager. There are four types of scene transition: regular, fade, reload, and game load.
    - Regular and fade are both similar, with fade having a fade to black.
    - Reload is used exclusively within the game scene to restart the level; it loads the new scene, disables its root objects, unloads the current scene, then activates the new objects.
    - Game load is more complex and is used when loading into the game from the main menu. It does a fade transition to a loading scene, loads the game scene in the background, and fades into it after the player presses A.
- The INIT scene is the persistent scene that is loaded when the game starts and contains many of the core system managers.
- Other scenes include splash, start, menu, ship, character, track, loading, game, and summary.
- All information collected as scenes are progressed through is stored in flags in the GameManager. These include selected ship, character slots in use, and level selected.

10.0 Levels

- A level in the typical sense consists of two parts: the ship and the level.
- The ship determines the max number of players and contains the area that players can move within. Each ship has different layouts and station positions.

- The level is randomly generated using one of three difficulty presets: easy, medium, and hard. The presets determine the levels length and the number and length of events across the level, as well as various values for stations such as start fuel and refuel amount.
- Players win by reaching the end of the level.
- Players lose when the ships oxygen level reaches 0.
- If the player selects endless in the difficulty select, an alternative game mode is used. Instead of the player ship reaching the end of a level, events are continuously generated and are made longer and more difficult as the game goes on.

11.0 Items

- Grabbable objects are a type of interactable that can be picked up and dropped. They are used to complete other interactions.
- Fuel: A grabbable object produced by the reactor. Used to fill fuel tanks on the engine and turret.

12.0 Interface

12.1 Menu

- There is a splash screen that requires no player input to progress.
- A 'Start' screen loads after the splash screen. Pressing start will progress the game to the main menu and store the controller that triggered it as the primary input (P1).
- The main menu consists of:
    - Play Button –Play button will load the Ship selection screen.
    - Options Button – Options button will enable a sub-menu.
    - Exit Button – Exit button will quit the game.
- The options sub-menu consists of:
    - Music volume slider.
- Ship Selection – The ship selection screen consists of a sprite swapper. Swapping ship sprites requires pressing left and right on the primary controller's DPad. Pressing A after selecting the desired ship will confirm selection, register the selection with the GameManager, and cause the Character Selection screen to load.
- Character Selection – Depending on what the max players count is on the selected ship, that many controllers can be connected and can join the game. For example, if a medium map with a max of 3 players was chosen and there are 4 controllers connected, only three player slots will be shown. The first will be populated by the primary player/controller. The other two will show a "Join" prompt. If a controller is disconnected that is not joined, the corresponding joinable slot will be removed. If one is added, and the max players are not reached, a new slot will appear. If a user joins a slot, and then disconnect their controller, their slot will persist and show "Reconnect Controller". When all connected players are ready, the level select scene will load.
- Level Selection: Determines difficulty level.

12.2 UI/HUD

- In the bottom right of the screen is the scanner and the ship's oxygen level.
- The scanner consists of a bar representing the level, a flashing icon representing the position of the ship along the level, and event icons that are displayed when the

scanner station is used. When the scanner is activated, an image will travel from the ship icon to a set distance ahead of it, with events it passes becoming visible for a short time.

- The scanner background will also become highlighted the area of an event when it is active and when the 'ping' icon is within an event. The color of the highlight is dependent on the event type.
- The ship's oxygen level is effectively a health bar for the ship. It will go down while the ship has holes in it until it reaches 0 and the game is lost.
- Asteroid indicators are displayed at the edges of the screen to show where the asteroid will come from and its direction.
- Interaction prompts serve multiple purposes and are displayed above interactables. When a player is not in range of the interactable, an icon is shown to display information on the type of interaction e.g., fuel, repair, etc. When a player can interact, the icon is swapped to show the button used.

13.0 File and Programming Naming Standards

13.1 Folder Structuring

- Art: Stores all sprites.
- Models: Stores all meshes.
- Materials: Stores all materials.
- Scripts: Stores all .cs files
- Prefabs: Stores all prefabbed objects.
- Audio: Stores all sound files.
- Documents: Stores all project documents.
- Animations: Stores all animations.

13.2 Coding Standards

- Public member fields: PascalCase
- Public static fields: PascalCase
- Public global fields: PascalCase
- Private member fields: m_MemberField
- Private static fields: s_StaticField
- Private global fields: g_GlobalField
- Method: Method()
- Method argument: a_Argument
- Class/Struct: PascalCase
- Local variables: camelCase
- Spaces between brackets and declarations. Examples:
    - for (int i = 0; i < 10; ++i)
    - void Foo(int a_Index, string a_String)
    - Get< string >()

14.0 Asset List

File Name	File Type	Description of use

15.0 Technical Risks

- Over-engineered code: Overdesigning code to tackle open-ended requirements can lead to long development times. This can manifest creating useless function overloads for every requirement.
- Inconsistent code design: Competing systems can lead to spaghetti code, and logic errors.
- Changing game design: This can cause some parts of code to become redundant, and other parts to become incompatible, leading to complete redesigns.