Functional Specs.

Outlier

GAM 300/400 – Fall 2017

Table of Contents

1. INTRODUCTION, Overview and Purpose

Outlier is a suspenseful first-person 3D puzzle game in which the player uses terminals to hack into a machine-controlled world to escape from the Hive Mind and its minions.

Outlier will be available through a stand-alone installer that can be run on any PC using Windows 7 or better.

1.1 Outcomes and Scope

The planned outcome of *Outlier* is a stand-alone stealth based puzzle game, in which the player must avoid capture by hive-mind based AI. *Outlier's* custom engine will make use of a PBR (Physically Based Rendering) pipeline created in Vulkan.

- *Outlier* will be prototyped in Unreal, which will then be ported and continued to be developed using Maya and a custom level editor near the end of October 2017.
- *Outlier* is planned to take place within an eerie suburban environment.
- An abandoned factory is a stretch goal environment for the game.
- Each level will contain 1 or 2 puzzles, lasting around 5-15 minutes each.
- Players will need to balance speed and patience in order to hack into objects around the environment while evading capture by the hive mind AIs.
- The hive mind AIs will share information across a network, simulating a hive mind. Should one enemy see the player, all AIs will be alerted to the players presence and location.
- Using Wwise, the game will feature data-driven dynamic sound and music to flow with the current on-screen action (or lack thereof).
- The story of the world will be revealed environmentally instead of explicitly through narrative.

1.2 Project Requirements

- The final product should be installable on any Windows 7 PC or better, exact computer specs are TBD.
- *Outlier* can be played with either a mouse and keyboard or a gamepad.
- TCRs as set out for GAM300 will be met.

1.3 Development Requirements

- The development environment for this project is Visual Studio 2017, on any Windows operating system 7 or better.
- Unreal 4.0 will be used for prototyping while the custom engine is under development.
- Art assets will fit in the style of Simon Stålenhag. 3D models will be created using Maya or Blender. 2D textures for models and terrain will be created in Photoshop or

1.4 General Constraints

- The final product must work on a standard Windows 7 PC.
- The final product should support Windows 8, 8.1, and 10 operating systems.
- The final product should install all necessary software such as drivers, dlls, etc.
- The final product should be stand-alone, in that no middleware is required to run.
    - If any middleware is required, the installer should handle it's installation.
- The final product will meet all the TCRs currently required for the GAM 300/350 courses.

2. FUNCTIONAL DESCRIPTION

2.1 Install and Uninstall

Outlier should be able to be installed on any Windows 7 machine with the appropriate hard drive space and other system requirements. An error message should be generated if the project is unable to operate on the machine.

The installer will be simple to use, creating it's own directory within the DigiPen folder in ProgramFiles(x86) as per GAM 300/350 requirements. Saved data will be saved under a custom directory in the MyDocuments folder as per GAM 300/350 requirements.

The installer will be a full installer as per DigiPen requirements.

Uninstalling the game will remove all of the installed elements of the project. The saved data for the game in the MyDocuments folder may be left behind (TBD).

2.2 Game Mechanics

Hacking

The core mechanic of the game, players will have the ability to hack into terminals found within the many different puzzles in *Outlier* in order to progress through the harsh and mysterious world. Through a terminal, players will have access to security cameras. These cameras will allow the player to "focus" on various interactable objects and enemies in order to gain partial control over them.

Hacking will be a duel-edge sword however. While in this state, the player will be slowly alerting any nearby AIs to their presence and precise location. Players will need to think quick and plan ahead in regards to what they wish to achieve in order to both interfere with the Hive Mind's totalitarian control, while avoiding capture.

Avoidance

In order to survive in the mechanical and abnormal world, players will have to focus on remaining out of sight and reach of the Hive Mind's members. From humanoid automatons to towering three-legged tripoidal horrors, remaining hidden in the shadows will be a key feature to succeeding and completing the many challenges within the game. As making progress through the game will require the player to alert the Hive Mind of their presence, players will need to master speed and patience in order to survive.

Health

In order to push the themes of being weak and powerless, the player will not have any health stat. Instead, the player will have to stay out of reach from the Hive Mind's mechanical minions in order to evade capture and re-assimilation. Should the player ever be caught by one of the game's enemies, they will respawn at the last checkpoint.

Death/Checkpoints

Players will be able to die within the game's later half, as this is when the Hive Mind will dispatch its mechanical minions to track down and halt the player's progress. As players "die" after being caught by any subordinate of the Hive Mind, checkpoints will be placed after each puzzle to ensure progress is maintained. Checkpoints may also be present in longer puzzles after a significant feat is achieved. This will be indicated by an on-screen icon.

Hive Mind AI

The primary antagonist of the game, players will need to learn and understand the patterns and behavior of the Hive Mind and it's many members in order to progress and evade capture. Sharing a network of information, should one enemy be alerted to the player, the entire network will enter its alert state, hunting down the player like prey.

2.2.1 Player and Character Functionality

- Player – A bipedal robot. Despite being a first-person game, the player will still need a model in order to be seen through security camera footage. Animations for the player from the first-person view will be created through a view model (TBD), while the animations and model for the player from third-person view will be shared with the Astrobyte enemy (recolored).

Event	Sound Effect	Description	File format
Idle	Mechanical	Idle animations that would be randomly cycled through in order to create an interesting standing pose. Player should look "on-edge" to fit the theme.	3D
Walking/Running	Mechanical Footsteps	Walk/Run animation accompanied by SFX of player footsteps.	3D
Hacking into terminal	Electrical Zap/Sci-Fi	An electrical current shooting from the player's "hacking device" into the terminal, granting access. Animation, SFX, and particle effect.	3D
Captured	Powering Down	Slight struggle animation, screen slowly fading out, light shutting down from player's "eyes".	3D

- Astrobyte – Similar in look to player due to being the same type of robot, but with a magenta color/lighting scheme emitted through the eyes of the robot. This enemy is a bipedal creature, using the same rig and animations of the player.

Event	Sound Effect	Description	File format
Idle	Mechanical	Idle animations of the robot looking about predatorially.	3D
Walking/Running	Mechanical Footsteps	Walk/Run animation accompanied by SFX of player footsteps. Should share SFX of the player due to narrative and thematic tones desired.s	3D
Stunned	Sparks	Light in the eyes of the Astrobyte interperlates quickly off. Sparking particles from the robot (possibly smoke particles as well). Eventually reawakens without indicating awareness of ever being stunned.	3D

- Gila – Monstrous tripoidal creatures maintaining control over a set amount of Astrobytes, these horrors appear to act as protectors to the Hive Mind. One Glia acts as the primary antagonist, hunting down the Player through the latter half of the game

Event	Sound Effect	Description	File format
Idle	Alien Machinery	Idle animations cycled through of the Tripod's "eyes" looking about. Legs are static. Searchlights may be moved to simulate looking about via code.	3D
Walking	Mechanical Boom	Heavy impact of legs moving about towards a designated target point. Possibly leave crater decal where legs hit the ground to show cracks in the earth. Slow.	3D
Destroyed	Sparking, Mechanical "screaming"	Visuals of spotlights flickering off along with "eyes". Sparks coming from main body of the tripod. Eventual explosion and gibbing of the tripod.	3D

- S3M1 –Tiny robotic creatures seemingly used to fulfill any minor role the hive mind needs. These mechanical bots are dispatched infrequently to search for the player when the larger, more purposed machines were failing at their task. Not fully connected to the Hive Mind, these rat-like critters can be easily hacked into by the Rouge to use directly against the Hive Mind by widening the player's vision. Unfortunately, due to their unfinished nature, once hacked they tend to self-destruct and fall apart after a short period of time.

Event	Sound Effect	Description	File format
Movement	Wheels Spinning	Simple animation of a wheel rotating under the base of the robot.	3D
Player Sighted	Powering Up SFX	Eyes slowly interpolate from "idle" color to "alerted" color. SFX helps further signify state of robot.	N/A
Found Player	Alarm SFX	Alarm is emitted from the robot, robot stops moving, eventually exploding after several moments.	N/A
Explode	Explosion	S3M1 gibs into various parts with a smoke trail left behind. Possible decal of a scorch where the S3M1 once was.	N/A

2.2.2 Artificial Intelligence Overview

The AIs in the game will be hierarchical, as follows:

1. UNIT PATHFINDING – This will allow each unit to pathfind to a predetermined destination.
2. UNIT TASKS – this AI will determine the best destination or objective for each units.

2.2.2.1 Functional Development of AIs

The AIs will be data driven, using XML or other data storage. Throughout the development cycle, several templates will be created by the programming team. These templates can be reused by end-users.

- FILE FORMAT: needs to support different versions, whether through loading or pre-loading/pre-processing.
- NAMING CONVENTION: needs a solid means of identifying different types of AIs for use within the game.
- AI SELECTION: loading an AI into the game during runtime should be easy: some form of a pull-down menu or other selection that lists all AI file types within the program or user directory. Naming convention then becomes critical.
- TEMPLATES: it should be easy to create a basic AI and then save it off as a new variation with minor changes. Changes to the original AI should NOT trickle down to the variations.
    - Implemented via the same archetyping system as the rest of the engine
- Types of AIS:
    - Patrolling
    - Tracking
    - Etc.

2.2.2.2 Addition of new levels or maps, NavMesh creation

Addition of new maps or levels should be simple. Any map or level within the appropriate directory should be something the engine could load.

- IN GAME LOADING: The engine should be capable of loading any level placed within the proper directory (maps) inside the main game's directory.s
- LOADING ERRORS: If the engine is unsuccessful in loading the proper file, the game should default to load the title screen to prevent a critical crash of the program. An error message will also be displayed to the player.
- NAVMESH GENERATION: Will be accomplished using Recast (3rd party library).

2.2.2 Development of Graphics

The custom engine's graphics will be handled through Vulkan. The game will use PBR in order to create a high-fidelity, modern look. Particle effects and lighting will be handled by technical artists in order to enhance the look of the world.

2.2.3 Development of Physics

This project will use the Bullet Physics Engine SDK to handle collision detection, forces, velocity, and other aspects of the game that require a sophisticated physics system.

As a stretch goal ropes may be created through the use of this SDK.

2.2.4 Development of Audio

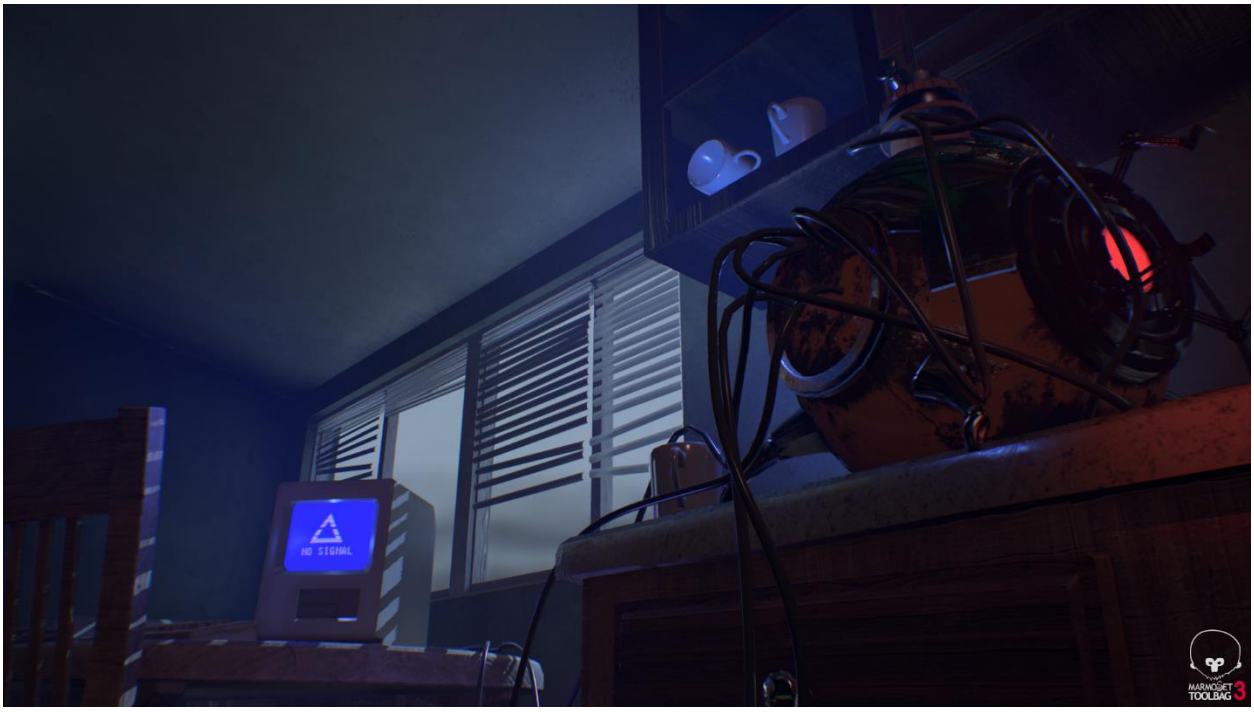
Soundscapes and audio effects will be played through Wwise audio middleware, with real-time game parameter input via C#/C++, and supplemental audio tools will play adaptive music. The game will support stereo signal at the standard 44.1kHz sample rate. Additionally, this project has a high probability of utilizing custom in-house DSP effects to modify audible content in real-time.

2.2.5 Development of Editor

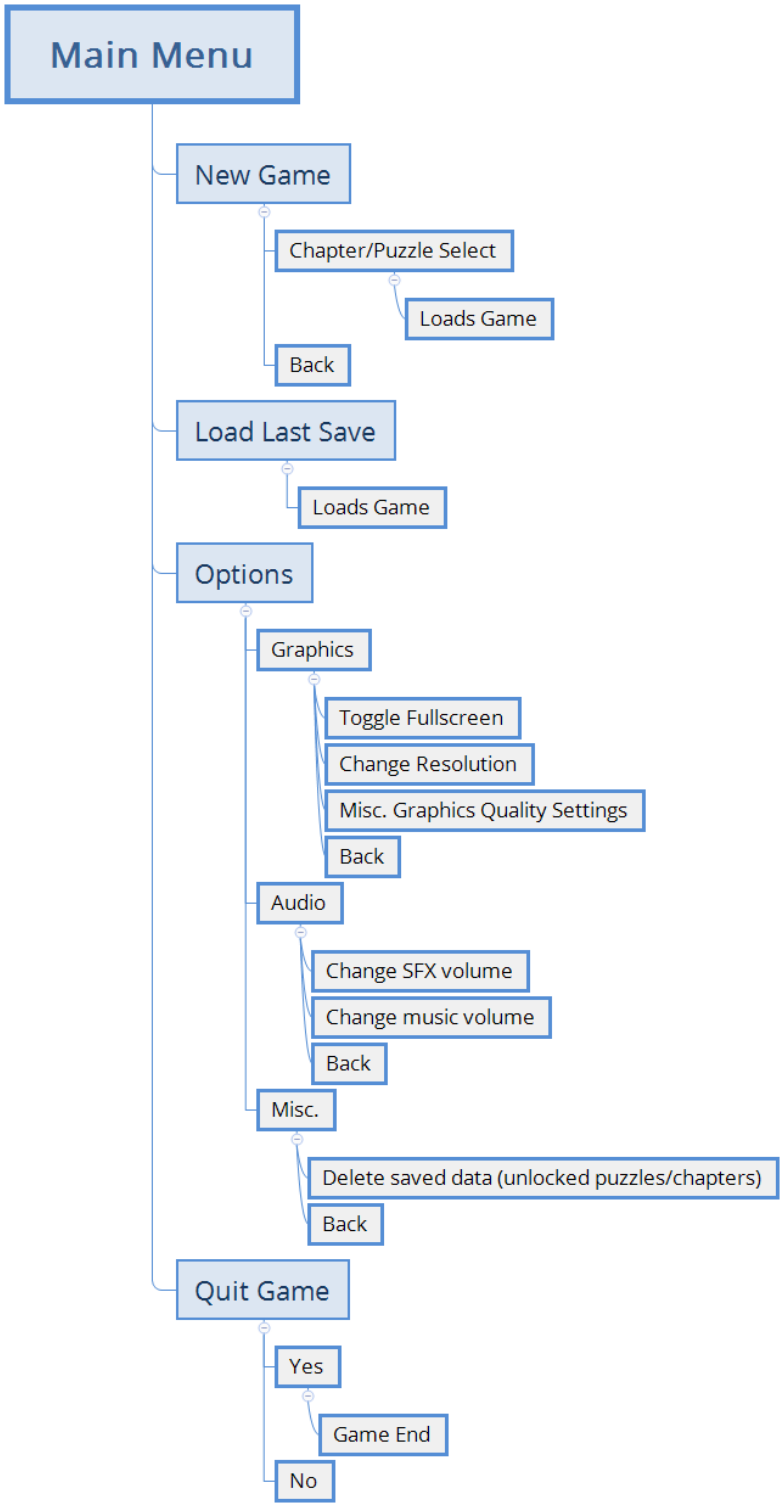
Our standalone editor will be written using WPF/C#, running the unmodified engine under the surface. This lets us develop tools for use either in a standalone editor setting using C#, or tooling that is available in game using Dear ImGui. The standalone editor will support docking viewports much in the vein of Visual Studio. The C# editor will be using the .Net P4API to interface with perforce to allow seamless checking out and checking in of level files. Binding C# for the purpose of editing will be done using C++/CLI, which will give us flexibility in the Engine/Editor interface. This will allow for full engine functionality to be available in the standalone editor.

2.2.6 Runtime menus and controllers

All in game interfaces and menus should be usable by either keyboard/mouse, or a gamepad. The game will not allow for custom binding of controls on either mouse or keyboard. After the tutorial of the game is over on-screen HUD prompts for controls may be minimized (TBD).



2.3 User Functionality



3. SPECIFIC REQUIREMENTS

3.1 External Interface Requirements

3.1.1 User Interfaces

a) Hardware Interfaces

   a. *Outlier* will support keyboard and mouse within the stand-alone operation.

   b. *Outlier* will support a single gamepad within the stand-alone operation.

b) Software Interfaces

   a. Menus – *Outlier* will have a main menu with options for full screen or windowed mode, sounds adjusting, and other graphical features as needed for performance enhancing on lower end machines.

      Flow chart of menus.

   b. Pause – Pauses the game, allowing access to various interfaces to change settings on the fly, or return back to the game title screen/exit application.

   c. Save/Load –The game will be capable of saving and loading a player's progress through a level, the current location of all objects, and the status and settings of each AI present in the world.

   d. Credits – Alphabetical listing of developers names displaying the role of the member in regards to accomplished tasks and representable work in game.

3.1.2 Frame Rate

A target of the game will be to maintain a frame rate of at least 60fps on a typical modern system.