Milestone 1

John Manard and Fiona Soetrisno

January 21, 2022

1 Design

One of our research goals is encouraging the use of underutilized play styles through in-game rewards. Our game will be designed in such a manner that several different play styles are possible. One example of this is risky versus safe play styles. If our director AI notices a player is sticking to safe, ranged options, the game will offer a room with increased risk in exchange for greater in-game rewards. Thus, the player is positively rewarded for engaging with and learning an infrequently used play style. Alternatively, with our system, we could perform the inverse through rewarding the playstyle the player already prefers or not rewarding any specific playstyle at all. Through our project, we are looking to test whether or not players' preferences are biased towards already used mechanics or ones they don't use as frequently.

Another major goal of our project is to make an exciting gameplay-driven game. We wanted to design mechanics that are engaging and fun to play, but also have a sense of challenge for the player. With our game being an action-combat game, we are spending considerable effort making the player movement and combat feel rewarding and inherently enjoyable to play with. We have spent the majority of the first three weeks brainstorming and prototyping mechanics for this purpose. Additionally, we wanted to make our mechanics novel and have real-time trade-offs between risk and reward.

Our core principles that guide our implementation include player agency, replayability, and fluid mechanics.

For player agency, we wanted to give the player a choice for what weapons they are using and which types of enemies they want to face. Players get to choose the upgrades that they want to get, and though the rooms are random, we wanted the player to get a preview of the types of enemies that they'll be facing.

One of our most important principles is having fluid movement mechanics. As mentioned previously, we want the combat and movement to feel good for the player since that is how the player directly interacts with the game. We want to make sure the interactions between different components, such as enemy AI, weapon types, and player movement, all work together seamlessly and are balanced.

We want to ensure that the game is replayable and has different events that happen on each run. As a rogue-like, we are randomizing combat encounters, enemy spawns, and potential upgrades. The randomization factor gives several combinations and possibilities that the player can experience. Additionally,

the different weapon types give players the option to have different play styles and focus on different aspects of the game each time they play it.

2 Background

The implementation of our game will be done in Unity. Unity is an engine for developing interactive graphical applications and has a large selection of teaching resources for students and developers. Implementing a renderer, animation system, physics system, and other needed features for a 3D game is out of scope for our senior project, so we chose Unity for its wide array of existing features. Unity includes a scene editor, where developers can visually move objects in the game and set up levels using a graphical interface. This scene editor rapidly increases the speed at which developers can playtest and update scenes for prototyping purposes. Unity includes all necessary core features for developing a 3D action game and allows programmers to utilize its various libraries through a C# scripting API. By calling and accessing the Unity API, developers are able to influence objects in the engine. This is done by using an entity component system, in which each object in the scene has modular scripts that can update the object based on the engine's core input loop. Developers design these scripts to utilize the built-in features of Unity in a manner that allows modularity and reuse.

One of our primary goals for this project is to quickly prototype engaging mechanics and systems given our timeline and experience. Unity has a wide variety of packages used for common, specific needs of small development teams that allow us as a group to quickly implement features such as character controls, camera movement, animation state machines, and animation blending. All of these features come as full packages for developers to quickly use and modify. Unity has extensive documentation for each of these packages and built-in features, exposing all public methods and variables in the online documentation. The documentation also features many examples and detailed descriptions of each publicly accessible member. Unity's detailed documentation allows us as programmers to quickly develop and understand the many components of the game engine.

There are some drawbacks to using Unity, but they are largely unimportant for our goals for this senior project. For one, games made in Unity have a relatively large build size in comparison to games developed with other engines and game frameworks. Since the devices we are building for have large

memory and storage specifications, the overhead of developing in a general engine like Unity is overshadowed by the speed of prototyping achieved with the engine and its packages. By using Unity as our engine, we also lose some control over the graphics pipeline without extensive work to make a scriptable pipeline. However, this is not needed as our game research and goals are focused around gameplay mechanics as opposed to visuals. Additionally, Unity provides a host of visual editors for vfx and particle systems for more general use cases, which will be utilized to a greater effect than custom render pipeline effects in our game.

The genre of game we are producing is a top-down action game set in a dungeon. In this genre, the player works to defeat enemies using fast paced weapons and strikes. The player needs to make moment-to-moment decisions to succeed in defeating the enemies of the game. These decisions usually involve a movement mechanic, like a dodge or a roll, that allows the player to alternate between dealing damage and avoiding damage. Players fight through each room, defeating all enemies in the room to progress forwards. Our research is going to design what is known as a director AI, which is an algorithm that creates encounters or enemies based on a metric it wants to optimize. Common uses for director AI are controlling pacing of games and increasing the believability of the world. Our background AI's heuristic it wants to optimize is the use of all play styles in the game.

3 Related Work

3.1 Revenge System from *Metal Gear Solid V*

The Revenge System or Enemy Preparedness in *Metal Gear Solid V: The Phantom Pain* is a game mechanic that increases difficulty by adding gear to some of the AI enemies that counters your play style [1]. For instance, if you have been completing missions at night, enemies will carry flashlights or night vision goggles. This mechanic is added to cause the player to explore other options for completing missions and to prevent one mechanic or play style from becoming too prevalent. Our approach is similar in that our overarching game system generates different enemies in response to user play style and weapon choice in order to deter the player from having a single prevalent play style. Where our approaches differ is that all of the modifications made to enemies using the Revenge System inherently increase the difficulty of

the enemy as opposed to its original form without the modification. Our proposed mechanic would not have an increased difficulty version of the enemy, but rather change out the enemy type for a different one that increases the effectiveness of another play style.

3.2 AI Director (AID) from *Left 4 Dead*

The director AI from *Left 4 Dead* is an AI that uses the current players' actions and involvement with the enemies of the game to maintain an unpredictable and exciting pace of action [2]. The AI itself is a state machine of phases of play and determines state machine transitions with each players' involvement with the game and skill. The AI director in *Left 4 Dead* works to engage the player through the spawning of different types and quantities of enemies with the purpose of maintaining an adequate level of stress, whether that be high or low stress. The AI director in *Left 4 Dead* is categorized as a Non-NPC Controlled Dynamic Difficulty Adjustment (DDA) to control pacing [1]. Our AI director has a different goal: to encourage the player to use underutilized play styles. As such, our AI director will use entirely different metrics for player modeling and different states for spawning enemies. Additionally, the purpose of the system is not to manage a level of stress, but to vary the experience of the player in a way that is not solely dependent on difficulty.

3.3 "The Anecdote Factory" from *Far Cry 3*

"The Anecdote Factory" is an AI director in *Far Cry 3* designed to produce interactive environments in an expansive open-world. Due to the limitations of systems in combination with the large game environment, only entities near the player can be updated on the CPU [3]. This system provides a way for the game to create organic player experiences without having to fully simulate every entity in the game. The system spawns encounters into the world, providing a sense of realism and a variety of content. Our director AI's purpose varies in that our game does not have the computational limitations of *Far Cry 3* nor is it the nature of the encounters that are necessarily varying. Our encounters do not vary in objective or location, and the director AI's intention is to encourage the use of different play styles rather than to produce a believable world. Our director AI's purpose overlaps in its intention to produce varying experiences for the player based on their current context.

3.4 *Hades's* Combat System

Hades is an action roguelite game in which the player clears combat encounters in order to increase the strength of their character over the run and foster relationships with NPCs in the game tied to the upgrade system. Our game will feature many similarities to *Hades*, including its animation-based combat style, dungeon-style floors, and branching paths. Where our game differs from *Hades* is its emphasis on multiple play styles that are accessible at all times during the game. In *Hades*, the player selects one weapon to use for a run and all upgrades are based on that weapon. In our game, all weapons are simultaneously accessible throughout a run, and the variation in each room depends on the player's interaction with the game's director AI system. Rather than having the player focus on improving one play style available to them, we will allow them to emphasize one weapon and then challenge them to use the others, as opposed to encouraging continuous use of one weapon.

References

[1] A. Zhadan, "Artificial intelligence adaptation in video games," 2018.

[2] T. Thompson, "In the directors chair: The ai of left 4 dead," Sep 2017.

[3] T. Thompson, "The definition of [artificial] insanity: The systemic ai of far cry," Oct 2017.