New God
Senior Project

Evan Springer
Professor April Grow
September 2023
Introduction

Artificial Intelligence is used in all sorts of ways in various types of media. AI is often applied to Interactive entertainment such as video games. One of the most important parts of designing an AI for a video game is considering how “intelligent” it should be for better user enjoyment. For instance, making the AI too simple can cause frustration and impatience in the player, and making the AI too advanced can result in unexpected side effects and/or the player not getting personal satisfaction from in-game accomplishments.

My goal for this project was to figure out what level of “intelligence” is ideal for a “management” type video game, where the gameplay is centered on managing several AI characters rather than controlling their own character. Because the gameplay almost entirely relies on player-AI interaction, a well-designed AI is necessary. New God has the player interacting with several characters in an undeveloped environment, with the goal of building a thriving community of AI characters.

Background

I chose to make New God for my project because I wanted to create the beginning of a polished software product that can eventually be published. Additionally, a video game project allows me to set flexible goals and visualize a minimum viable product. Furthermore, it utilizes many aspects important to interactive software, such as UI design, AI controllers, animation, art, etc. My goals for making a video game is further developing my skills as a software engineer and designer by creating my own assets, as well as learning new skills to use in future projects. The success of this project depends on the quality of my implementation of the several aspects of software design, most notably my AI implementation, which will all be determined by user testing.

For the development of my project I chose to use Unity, since I had some previous experience developing video games in Unity from my Interactive Entertainment Engineering course. Unity is a game engine that provides several useful built-in capabilities like object physics and AI navigation. It also doubles as an IDE for scene editing, with simple drag and drop functionalities and live object editing. It also provides extensive documentation on many of its features, which allows beginners like me to develop easier.

I also used Piskel, a pixel art drawing program, for drawing and animating most of my assets. Since the art style I was going for was simple, I could draw all of my own assets and import them into my game with ease.
Related Work

I took inspiration from many other video games in the creation of various aspects of this project. The biggest inspiration would be Don’t Starve, which is a 2D Open World Survival Game very similar to my own. Like my game, Don’t Starve is isometric and uses two-dimensional graphics for its characters, items, and environments. However, my game differs as the character is not directly controlled through player input. Instead, the character takes commands from a prompter navigated by the user.

Sons of the Forest is another Open World Survival Game I took inspiration from. In the game many structures are built by laying out a “ghost blueprint” which gets built as materials get added to the building. Additionally, you can command the AI companion in that game to do things with a “command builder” similar to what I implemented. My game differs from Sons of the Forest by having the main character being commanded by the prompts, as well as not showing a live construction of the structure until it has been completed.

The artistic style of black-and-white objects and stick figures was inspired by West of Loathing, a single player role-playing video game. My game is vastly different in genre, and the art style is pixelated and even more simplified than the style found in West of Loathing.

Design

The game I am creating is a 2D top-down/isometric video game centered on a small group of characters surviving in the wilderness. The core gameplay is centered on the player, who plays as a god giving commands to the in-game characters in order to ensure their survival, as well as developing their own abilities as a god. This game should appeal to audiences that enjoy open-world survival management games.
The art style of the game is pixelated line art, inspired by the game West of Loathing. This is so I can easily create plenty of assets on my own, while at the same time providing a clean and consistent look throughout the game environment.

Commands are made through a command builder, which is represented by a list of preset commands that are available. Specific details relating to certain commands are either filled in through another child list in the command builder, or selected in the actual world environment. Examples of commands would be “Go to V” or “Gather [...]”, where “V” indicates clicking a place in the video game world, and “[...]” indicates further command building. The character will then complete player-created commands automatically.

New structures, upgrades, and items can be unlocked through character actions that progress the game. The player, who plays the role of a god for the in-game player, can command the character to do certain things, such as crafting a tool, to allow for different resources to be gathered and progress in the game.

Implementation

The AI characters act as vessels to execute the player’s commands. They each have their own unique properties, such as commands and inventories, and act independently from each other. They navigate the game world on their own using Unity’s NavMesh tool. The player can give commands for the AI characters to execute by right clicking on the character and selecting a command, and then select further input through a command tree-like structure if applicable. Input (such as selecting an in-game object) is gathered by using Unity’s RayCast object that detects an object’s hitbox from where the mouse pointer was located. This sets the targeted objects as a destination for that character, which directs the chosen character to the desired object.

The command prompter was created with Unity’s UI canvas objects, and would appear/disappear based on if a character was selected/deselected, which was determined based on if the user right clicks on the character or not. Commands were selected using Unity’s button objects that executed a function if clicked. Commands available are “Go to”, “Harvest”, “Gather”, “Craft”, “Store”, “Get”, and “Build”, with all but “Go to” having secondary commands to specify the command further.

There is a wide variety of identical objects used in the game environment, so a Prefab of the object is used to easily duplicate each of the objects properties. Each interactable target has an Interaction class attached to it, and when the object is set as a target by a command, the object first waits for the character to arrive at the object. Then, a general interaction function is called. Since an interaction is unique among types of objects, each type of object implements a child class that overrides the interaction call with the correct interaction process for that object. Interactions entail anything from picking up an item to cutting down a tree. After this whole process, the character executes the next command in its “command queue”, if applicable.
Items are treated as normal objects in the scene, but are represented as a ScriptableObject in scripts. These items are held in inventories or storage units and are used in crafting and building other items and structures.

The player can also instruct the characters to build structures at a location the player designates. This structure checks for interference with other objects before being successfully placed down. Then, a “blueprint” of the structure is placed, and the selected character will automatically begin the building process by gathering any necessary items for the build and taking them to the blueprint. Once the necessary materials are added to the structure, it completes the project and can be interacted with as normal.

Analysis/Verification

The criteria for success requires satisfactory user responses when playing the game. In particular, I want their interaction with the AI to be as enjoyable as possible. If the AI characters need to be micromanaged or do not involve the player enough, the AI must be adjusted accordingly.

Due to how I implemented characters and objects, I can scale the project to have many more characters and objects without having to change core functionality. This means that the project can feature more gameplay and more user enjoyment in the future.

User feedback that came from playtests enjoyed the UI for the command prompters attached to each character, however they disliked how large the list of commands were. In later tests, playtesters appreciated the secondary prompters that condensed groups of similar commands into a separate menu. Overall, users found the interaction between the player and characters to be easy to learn and to extrapolate from.

Playtesters were divided on how the interaction between the character and the environment should work. One group liked how it was initially, where the desired character would be selected and then the player would manually select the target for the character. Another group wanted the manual selection of the target to be automatic. I was able to appease both of these groups by allowing both manual and automatic targeting with different commands, so that individual players can select the mode they want.

Most importantly, users suggested that items needed for certain tasks, such as an axe for harvesting trees, should be automatically collected from storage for the task, however the character should not automatically craft one. Many other specified “rules” for multi-tasked objectives were implemented based on player feedback.

Additionally, players desired a slightly transparent “ghost image” to show where a structure could be built. As an additional side effect of this feature, it also allowed for “incomplete” buildings that could be finished at a later time, which allows greater flexibility with building.
Future Work

The end goal of New God is to reach a publishable stage. While this prototype is far from that goal, the base game I have developed for this project is a good stepping stone to reaching that milestone. With consistent updates and frequent playtests, I believe New God can become a full application that is enjoyed by a variety of players.

As I get more user feedback I will make the AI more advanced and tailored to what the player base desires. For instance, playtesters want characters to get items from storages, but not automatically craft them. Tweaks in character behavior can be adapted to fit the user’s need, especially as the depth of the game increases. Furthermore, I will add even more objects and commands using the frameworks I have already created. This will flesh out the game more and hopefully make New God a more enjoyable experience.

Additional features I would like to add is character customization that differentiates the characters from each other more, a health and hunger system that adds complexity and higher stakes to the game, a brief tutorial that illustrates how to interact with the characters and the environment, and lastly some animals that can make the environment the characters live in more lively. These features would provide more incentive for the target audience to play the game, as it will make the game more challenging, fun, and custom tailored to the player’s liking.

Lastly, many visual elements can be tweaked to give the game a better presentation to the users. One such criticism many users shared was to resize all of the elements so that the stylized black borders would all be the same width, resulting in a consistent look between every object present in the environment. Additionally, some playtesters suggested minor sprite changes so that objects in the environment look more grounded rather than sitting in a white void, such as adding grass textures to the larger sprites.

Conclusion

While development of this project has been concluded by this time, I do plan to continue developing New God as a side project for the foreseeable future. I have accomplished all of the goals I wanted to, but there are many more goals to reach as development progresses. I have gained a lot of insight in how the user interacts with AI and what users expect from the AI. My familiarity with Unity’s development tools, such as the physics, UI canvases, animations, and sprite editing has greatly increased as a result of this project. This project has been a huge learning experience with several tools and design principles that will continue to help me design and develop other products in the future.
Bibliography


   https://store.steampowered.com/app/1326470/Sons_Of_The_Forest/
