Yulong Fan, Taylor Kroon, Zachary Zirkle, Daniel Arellano;

Title: Java Game Engine;

Mentor(s): Shuhui Yang, Computer Science.

Abstract:

The goal of our project was to develop a fully functional 2D video game engine, capable of allowing users to build their own video games using our user-friendly programmable game engine library. Game engines are tools used by game developers to quickly code games without developing them from the ground up. Our game engine software is developed using the Java programming language and addresses performance issues; such as memory and processing speed. Ensuring smooth gameplay, quick user response time, and functionality on all basic computers.

We developed our video game engine software system using the Java programming language, with careful consideration of memory and speed performance. Video games require thousands of data to be stored and processed in seconds. So, our software design approach was required to ensure that data does not take up too much memory space and allow for processing at a rate of thirty frames per second. Our game was developed in many iterations, developing each feature from scratch piece by piece. The software was organized into categories based on the functionality, to divide up the workload. Each team member was assigned a different piece of the video game engine software to work on. At the lowest level, the game engine is made up of two components, data processing and rendering. Rendering is when we print images to the screen. What images are printed to the screen is determined in the data processing portion of the software. The data processing handles the logic used for reading user input and the logic needed for in-game play such as collision detection, movement, animations and any other in-game interactions. The higher level components of the game include the entity, dialogue, item systems, non-playable character AI, path finding algorithm, combat system, and map builder.

Together all these pieces makeup what is our video game engine. Which is a user interface that allows persons to customize their own video game. All the background logic that prints images to the screen is encapsulated into our game class, no need for the user to worry about it. They may go straight into programming their own custom games. Our tile based map builder allows users to load in their own art assets and create their own environments to be loaded into the main game engine. Characters and items can be added and customized using our programming library. Users can also design their own dialogue and quest system using the built in dialogue logic system. The character AI gives life to our non-playable character, allowing them to move intelligently through the world detecting obstacles and remaining in user defined regions. The user's player character uses a path finding algorithm to avoid obstacles in the game environment.

In result, we have created a basic game engine capable of supporting an isometric tile-based RPG game. Where users may customize player animations, dialogue, quests, items, and more. Future implementations would include support for different video game types, such as side scrollers, and multiplayer games. Further optimizations to speed and memory could be made to improve the performance further, and also to improve on the ease of user usability of our programming library.