Avant-garde
PROJECT & DESIGN DOCUMENT

By Ruikang Lin & Jason Ng

Introduction 2
- Game Summary 2
- Inspiration 2
- Player Experience 3
- Platform 3
- Development Software 3
- Genre 3
- Target Audience 3

Concept **4**
- Gameplay overview 4
- Theme Interpretation (Sacrifice Is Strength) 4
- Primary Mechanics 4
- Secondary Mechanics 4

Art **5**
- Theme Interpretation 5
- Design 5

Audio **5**
- Music 5
- Sound Effects 5

Game Experience **5**
- UI 5
- Controls 5

Development Timeline **7**

Members **8**

Avant-garde

Introduction

Project Vision

Avant-garde is an action-packed, 5v5, real time strategy, and medieval-themed game where the 5 players manage a developing kingdom together and fight the other players' kingdom to take over the continent. In Avant-garde, each player plays an important role within the kingdom: the King, the Treasurer, the War Commander, the Scout, and the Missionary/Envoy. This forces the players to communicate strategies, to manage economy and workers, to direct and fight with troops, to explore unknown surroundings, and to expand their kingdom together. This also allows gamers who have different play styles to enjoy a game together.

Inspiration

Chess

Different chess pieces have different functions and powers to ultimately win the game. In (Game Name), there are 5 players each representing the King, the Queen, the Bishop, the Knight, and the Rook.

Warcraft 3

A real time strategy and 1v1v1 free for all game where players develop their kingdom, gather limited resources, and control their army to fight in a free for all match.

League of Legends

A 5v5 game where players work together to make macro and micro decisions and strategies to impact their chances of winning.

Player Experience

In a fair and randomized map, players in first person perspective must work in a team to ultimately defeat the other team. The players will explore their role in the kingdom and make important macro decisions and micro skills to leverage their change of winning. The players must communicate and use their knowledge and information they have obtained on their surroundings and their enemy to craft a clever strategy in order to defeat the other team.

Platform

The game is developed to be released on Windows, MacOS, and Linux.

Development Software

- Vulkan, GLFW, GLM (Graphics APIs)
- C++
- CMake (executable)
- Golang, Docker (Server)
- Nginx (load balancer)

Genre

Multiplayer, Strategy, Action, 5v5, RTS, Role-Playing

Target Audience

Without too heavy or too complicated ideas, simple micro mechanics, and complex macro mechanics, this game is marketed to **casual game players** who enjoy role-playing and exploring as well as **competitive players** who enjoy creating innovative strategies and managing micro controls.

Development Timeline

September

- Complete Project Proposal
- Have a rough understanding about our game
- Set up GitHub for the project
- Set up simple CMake file to run the project for Windows, MacOS, and Linux, ensures easy compatibility without long set up for new developers

October

- Go through Khronos Vulkan Tutorial
- Make a triangle display on the executable
- Add simple user inputs and user interfaces
- Start to research, document, and work on physics engine
    - Game engine architecture
    - Real time collision detection
- Start to research, document, and work on graphics engine
    - 3D math primer
- Start to research, document, and work on data workflow and management
    - Game resource/economy management
    - APIs
- Start to research, document, and work on server-client connection
    - Docker, Golang, UDP

November

- Adjust and improve CMake file if needed after the tutorial
- Continue on research, document, and work on physics engine
- Continue on research, document, and work on graphics engine
- Start to research, document, and work on a world editor
- Start on the poster

December

- Finish the poster and print it
- Wrap up what is currently done for the project and document it in Github for next semester

Members

Ruikang Lin (Project Lead)

- C++ and CMake, Windows and MacOS implementations (MoltenVK)
- Designing user interfaces
- Looking into user input and windows management with GLFW
- Looking into game engine architecture
- Looking into real time collision detection
- Looking into world editor
- Start to define space and dimension with data structures for world generation and player movement within the world
- Integrate data structures for display with graphics and basic player movements

Jason Ng (Co-lead, Arch-away)

- C++ and CMake, Windows and Linux implementations
- Setting up the Vulkan Project
- Designing textures/models to be used
- Learning 3d rendering, raytracing, reflections, etc. with GLM
- Learning how to implement the above
- Learning basic graphics optimizations
- Designing the project such that it is easy to create objects with physics in the Vulkan API

David Hao (Backend, NYU Student)

- Work on game design and data management
- Design a game resource workflow within the world
- Looking into implementing a server with UDP connection for multiplayer using C++
- Dockerizing the server with Golang (Alpine)
- Writing APIs for sending data between frontend and backend