SiMSU
BROWN UNIVERSITY EDITION

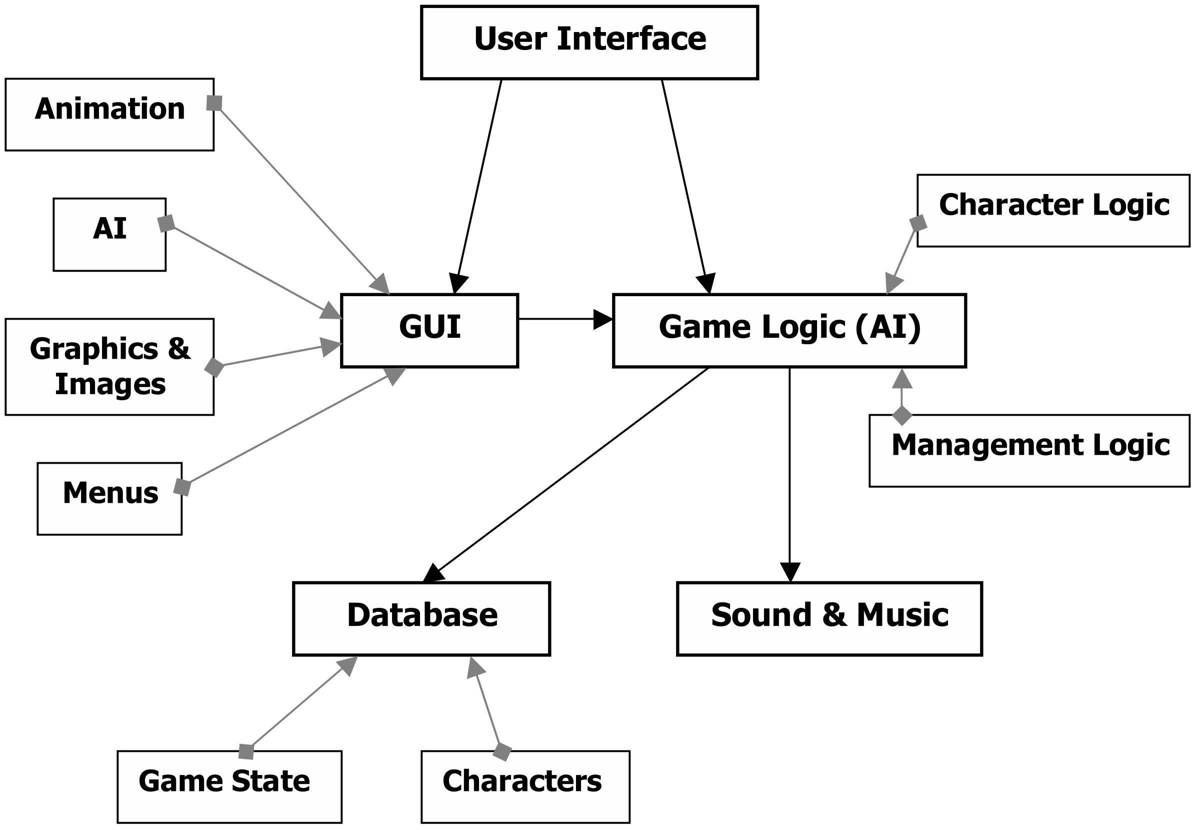
Top Level Design Document

Michelle Lin

Introduction

This design is based upon Rich Hsieh's idea and project, SimsU: Brown University Edition. Based upon the original requirements, specifications, and system model diagram, I have developed a top level design. This design is very similar to Rich's design, with only a couple of additions. The original design was clear and well-laid out, so there are no major changes.

Component Diagram



User Interface

- This is where the user interacts with the game.
- Graphical support through the GUI and actions interpreted by the Game Logic
- Mouse and keyboard support

GUI

- Top-down view of the campus using an isometric tile-based map.
- Interface for manipulating buildings, people, and the map.
- Menu support to allow user to set up game by selecting items to place on the campus, selecting student types, loading and saving games.
- Controls animation of buildings and the activities inside of them.
- Contains all images and graphics for the game, i.e. buildings, people.
- Animates the characters in the game by allowing them to walk, run, make gestures.
- The AI determines whether certain moves are allowed in the building process.

Game Logic (AI)

- Controls the main aspects of player control in the game and decides the results of decisions the player makes.
- Management Logic determines how certain buildings, classrooms, and professor hires will affect how the game is played.
- It also keeps track game play information and statistical information. This includes the status of buildings, condition of the university & its student body, financial situation, tuition requirements, total student enrollment, and number of professors.
- The Character Logic controls how characters will react to certain people and situations based on the information stored about the character in the database.
- Given the character's profile, will determine the mood.

Database

- Stores information about the game state, such as number and types of buildings, number of enrolled students, faculty, financial information
- Stores information about each character in the game, including where character lives, classes taking, tuition requirements, social life, and happiness.
- Allows for loading/saving of files.
- Information stored here is used by the Game Logic to implement AI characters.

Sound & Music

- Provides the audio effects when certain effects occur.
- Background music for the game.

External Dependencies

A critical part of this program is the graphics and animation. This relies upon OpenGL. The artwork used in the program can either be obtained from existing graphics, or rely upon a group member to create them.

For sound and music, an mp3 player may be used, which would create another dependency.

Group Organization & Task Breakdown

Project Manager: Responsible for keeping the project and all group members on task. Monitors the group's progress and creates and adjusts project deadlines. Coordinates communication within the group so that all members know what is going on with project. Schedules meetings and pays close attention to group members and their contributions to the project, as well as pays attention to concerns of group members.

Librarian: Maintains documentation throughout entire project. Updates any changes in specifications or project design. Creates user manual and documentation. Creates Makefiles.

Architect: Responsible for the design of the project. Will know the project design intimately, and know all the details and about the design for each piece of the project. Responsible for making sure design is logical and well planned out. Approves design changes.

Coders: GUI (2 people), Database (1), AI (1-2), Game Logic (2), Sound/Music (1). These members will be responsible for writing the code to make the program functional. They will design and test their individual parts. Some of the coding portions may be smaller than others. For these members, they may assist another person on other parts of the coding or may have non-coding tasks associated with the project.

Testing Coordinator: There will be tester who is responsible for coming up with specific test cases. The tester will also search for external users who would test the program.

Schedule

3/7 – Final Design

- System model diagram and high-level design completed.
- Create proposals for graphical user interfaces and create artwork.

3/12 – Interface Proposals

- Component level testing strategy.
- External interfaces for communication between other modules.
- GUI layout should be completed.

3/21 – Final Interfaces

- Final interfaces created based upon user evaluations and comments from group.
- Clear understanding of flow of control in program.

4/4 – Detailed Design

- Design for each module and the components of the module should be complete.
- Design freeze and begin intensive coding.
- Documentation needs to be updated with all changes made.

4/16 – Initial Integration

- Individual modules near completion with testing of these components begun.
- User interface should be completed, and integration between all modules begun.

4/23 – Implementation completed & Testing, Debugging begun

- Final project implementation should be completed.
- No more adding to the system of new parts.
- Extensive testing and debugging begin now.
- Have external users test the program.
- Integration should be top priority.

4/28 – In-class Demo

- Integration should be finished. Now major task is checking for bugs that may have occurred during the integration process.
- Still debugging and testing individual parts and the system as a whole.
- Major bugs should already have been discovered.

5/2 – Public Demo

- Continue testing and debugging. Project should be nearing final stages of completion.

5/9 – Final Demo

- Debugging and documentation completed.
- Final project handed in and should be completely functional, meeting all requirements laid out in documentation and specifications.

Assumptions/Modifications/Misc.

The modifications I made to the system model diagram was to clarify the functionality of the GUI on the diagram. The description in the original specifications were clear, but not as clearly laid out in the diagram. I added components which are contained within the GUI. I also included an AI component into the GUI. In the original specs, the AI for building to make sure building changes were legal was done my the management logic. I felt that it would fit in better if it were a part of the GUI.

In the database portion of the diagram, I included two components to clarify the distinctness of the game state information that needs to be stored, as well as the character information. From what I understand, you are able to control any character in the game. This means you'll need a database which stores the current data of every character at all times. There isn't a set maximum in the specs, so this number could be extremely large which would result in a huge amount of data being stored. This would therefore be separated from the game state information.

Part of this game involves determining the "happiness" of characters. It wasn't laid out in the specs how this would be determined. An algorithm or formula will need to be created allowing for this calculation of a character's mood.

Otherwise, everything else was clear and no major changes were made.