Requirements

ENG1 Team 9

Jacob Dicken
Bertie Cartwright
William Croft
James Dovener
Henry Chan

Introduction to Requirements

To begin eliciting the requirements we first analysed the product brief, which provided us with some initial fundamental requirements of the game.
We also noted down any remaining questions we had around the product brief, and then created a list of important questions to ask the stakeholder, ordered by priority. These were questions such as the role of sound effects in our game or guidance on the art style/theme we should choose.
Additionally we did some further research into sensible system requirements for the game (e.g system specs and supported OS's). From this, we found that devices with at least 2GB of RAM and 2 or more CPU cores should be sufficient for a simple 2D game.

Next we set up a meeting with the client to discuss the questions we had curated. In doing so we gathered and negotiated new requirements and cleared up any ambiguity in the requirements we had already gathered.
The client also noted that we had a lot of freedom to decide how some of the lower level design elements of the game worked, along with the style/theme of the game. This reduced the number of requirements relating to the style and design of the game.

During the meeting we also worked with the client to come up with a SSON which will help the whole team understand the broad concept of the game.

Single Statement of Need (SSON)
The system will consist of a campus planning game, which allows the user to place buildings, and simulates an in-game time of 3 years within 5 minutes of real world time.

After collecting the requirements we needed a way to present them in a way that was easy to edit and reference. To accomplish this we created tables to house the requirements. Firstly we made a user requirements table which holds all the requirements that were obtained from the meeting with the client. These user requirements were written in a non-technical way which allows for anyone involved in the project to understand their meaning and have a corresponding priority of either shall, should or may - this provides a clear indication of the priority assigned to a given requirement.

Next we made 3 tables for system requirements: functional, non-functional and constraint. The functional and non-functional requirements were determined by examining the user requirements and then creating descriptions of how the system will satisfy these requirements. Functional is for things a system should do and non-functional is for qualities a system should have. The constraint requirements were determined from the initial research we did into sensible system requirements before the meeting.

Every requirement in each table has a description of what the requirement is and also an ID. The ID is a meaningful name (for example UR_BUILDING_COUNTER) which is prefixed with either: UR, FR, NFR, or CR which correspond to: User, Functional, Non-functional and Constraint requirement. This allows for easy identification of what type of requirement the ID corresponds to but also allows for easy referencing in other areas of the project.

User Requirements

ID	Description	Priority
UR_CAMPUS_CREATION	The user shall be able to create their own university on a provided map which includes obstacles where buildings cannot be built.	Shall
UR_BUILDING_VARIETY	The user shall have a number of different building types to choose from. The variety of buildings should include at least: one place to sleep, one place to learn, one place to eat, and one for recreational activities. Users may also be able to upgrade their buildings.	Shall
UR_EVENTS	The user shall be able to interact and react to events that occur during the course of the game, these depend on the difficulty setting chosen by the user.	May
UR_STUDENT_SATISFACTION	The user can increase or decrease the satisfaction levels of the students in a variety of ways (for example having entertainment and food buildings near accommodation, and reacting to events appropriately).	May
UR_GAME_PROGRESS	The game should simulate a time of around 3 years within a time frame of 5 minutes. And have a counter on the screen that displays how much time is left. When the timer stops the game should end.	Shall
UR_IMMERSION	The game should be immersive for the player. Meaning that sounds and graphics should match the theme of the game.	Should
UR_TARGET_MARKET	The game should be suitable to be played by 16-20 year old students, or people who want to be students.	Should
UR_DIFFICULTY_SETTINGS	The game may contain multiple difficulty settings that the user can choose from. These settings should affect different parts of the game (for example increase the frequency and difficulty of events and change the amount of money you start with). The default setting should be the easiest which helps with accessibility for new users.	May
UR_PERFORMANCE	The game should perform well on the minimum spec machines and provide a pleasant user experience.	Shall
UR_INTUITION	The game should be intuitive to play at the fundamental level, for example, it should be straightforward to select/place buildings and deal with events. The game should stay intuitive for users with no gaming experience. Also the game should react to the users inputs.	Should
UR_BUILDING_COUNTER	The game shall have a counter denoting how many of each type of building have been placed so far.	Shall
UR_MONEY	The game may include an in-game currency system that is used to purchase buildings. The income rate of this currency should increase accordingly with student satisfaction.	May
UR_MAINTAINABILITY	The game should be easily modified and maintained by future developers	Shall

System Requirements

Functional Requirements

ID	Description	User Requirements
FR_TIME_LIMIT	When the time left on the game timer is less than or equal to 0 seconds the game must end, stopping the user from doing any more actions and displaying their student satisfaction score.	UR_GAME_PROGRESS
FR_MAP	The game shall provide a visual campus map for the user.	UR_CAMPUS_CREATION
FR_BUILDING	The game shall allow users to place/build and possibly upgrade buildings on the campus map. The time for buildings to build or upgrade should be dependent on the type of building and the difficulty setting.	UR_CAMPUS_CREATION UR_DIFFICULTY_SETTINGS UR_BUILDING_VARIETY
FR_BUILDING_TYPES	The game shall have at least one building of every type namely: Educational, Recreational, Residential and Eatery.	UR_BUILDING_VARIETY
FR_SATISFACTION_BUILDINGS	The number of each building type shall increase satisfaction. The closer each type of building is to another type of building should also increase satisfaction.	UR_STUDENT_SATISFACTION
FR_OBSTACLES	The map shall have preplaced obstacles that block the user from building on top of them.	UR_CAMPUS_CREATION
FR_EVENT_TYPES	Events should come in 3 different types: positive(Which benefits the player), negative(Which hinders the player) and neutral(Which does not affect the game)	UR_EVENTS
FR_EVENT_VARIETY	Events should occur throughout the game - these should cover a wide variety of events and consequences. These events will be set and occur at set times on the lowest difficulty but be randomised on higher difficulties.	UR_EVENTS UR_DIFFICULTY_SETTINGS
FR_DIFFICULTY_SELECTION	The game shall allow the user to select the difficulty of a new game. The default should be the easiest setting.	UR_DIFFICULTY_SETTINGS UR_INTUITION
FR_DIFFICULTY_EFFECTS	The difficulty selected should affect the variety, frequency of events along with the time to build and upgrade buildings.	UR_DIFFICULTY_SETTINGS
FR_TIMER	The game should have a timer that is displayed at all times during gameplay, showing the remaining time from the original 5 minutes.	UR_GAME_PROGRESS
FR_BUILDING_COUNTER	The game should display a counter at all times during gameplay that shows how many of each building type have been placed in the world.	UR_BUILDING_COUNTER
FR_USER_INTERFACE	The UI should display only important information to the user so that it is not overly complex and overwhelming.	UR_INTUITION
FR_INTERACTIVE_ELEMENTS	Interactive elements should react when clicked or moved (for example when a button is pressed its colour changes or a sound is made).	UR_INTUITION
FR_MONEY	Players should start with a predetermined amount of money based on difficulty and receive an income which increases proportionally to the student satisfaction level.	UR_MONEY UR_DIFFICULTY_SETTINGS
FR_BUYING	The game should allow users to spend their in-game money on purchasing and upgrading buildings and never allow users to purchase or upgrade a building if they do not have enough in-game money.	UR_MONEY

Non-functional Requirements

ID	Description	User Requirements	Fit Criteria
NFR_ERROR_MESSAGES	Any errors or warnings generated by the game should be easy to understand and not overly technical	UR_MAINTAINABILITY	The error messages should be understood by users who have no knowledge of the code.
NFR_PERFORMANCE	The game should run smoothly, at a framerate around 60, without hitches at any point during gameplay. This should apply on minimum spec machines.	UR_PERFORMANCE	The 60-second average frame-rate should exceed 58. The lowest 1% of frame times should also be above 50.
NFR_INTERACTIVE_ELEMENTS_REACTION	Interactive elements (for example buttons) should react quickly to user use. Preferably with some signal to the user like a sound or colour change.	UR_INTUITION	Interactive elements must react within <1 second
NFR_OPERABILITY	The game should be easily playable and navigable by new players.	UR_INTUITION	A new player shall be comfortable with the game after around 2 minutes of use.
NFR_DOCUMENTATION	The game should come with clear documentation that explains what each part of the code does and how to modify it.	UR_MAINTAINABILITY	The documentation should be understood by users who have no prior knowledge of the code.
NFR_END_OF_GAME	The game should immediately end after the timer has expired.	UR_GAME_PROGRESS	The game should end and block any further user actions within <1 after the timer stops.
NFR_IMMERSION	The game should have sounds and graphics that fit the theme of the game.	UR_IMMERSION	Sounds and graphics shall fit the tone of the game and not take away from the immersion of the user or stick out against the rest of the game.
NFR_THEME	The theme of the game should appeal to a 16-20 year old student audience.	UR_TARGET_MARKET	The theme should be something a 16-20 year old will relate to and be interested in.
NFR_CODE_MODULARITY	The code should be modular and easily extendable, without making the program difficult to follow.	UR_MAINTAINABILITY	Classes should hold references to necessary related objects and no more.

Constraint Requirements

ID	Description
CR_SYSTEM_COMPATIBILITY	The game must successfully build for Windows, macOS and Linux.
CR_MINIMUM_SPEC	The game shall run on 64-bit desktop computers with a minimum of 2GB RAM and a dual core CPU.