**Rooms**

*Group D*

# **1 Introduction**

 Imagine waking up to a normal day in your normal apartment. You get ready for work in the normal way, putting on your normal clothes, brushing your normal teeth, drinking some normal coffee, and in a normal & timely manner, you find yourself walking to the door of your apartment. Unsuspectingly, you open the door, step through, and hear it click behind you in its normal fashion. Suddenly, you stop. Your door has failed to lead you to the normal hallway of your normal apartment complex. Instead you find yourself in a room you have never seen before. Startled, you reflexively reach back to open the door you just came through, but to your astonishment it leads to yet another room you have never been to. Welcome to the universe of Rooms.

## *1.1 What is Rooms*

 Rooms is a top-down 2D exploration game with emphasis on plot, randomly generated levels. It offers players a unique world composed entirely of so-called rooms, apartment-like interiors filled with extraordinary content. The two goals of the game are survival and exploration. As players explore, they will encounter battles against monsters, but beware, not all are easy to find. Players can try to unlock the many secrets of the *Rooms* universe, and more importantly, how to survive it. Apart from the actual game Rooms features a room editor that will allow users to create in-game content which will be integrated into the game.

 Rooms is developed to run in the Google Chrome web-browser and is thus playable on any personal computer OS that supports the browser. We have also developed a website which will enable users to create and manage accounts, play the game and use the room editor.

*1.2 Game Concept*

Several key features in combination set this game apart from both oldies and modern games - an innovative level generation system, unique plot-based level ups and a dynamic light system integrated into the game play.

The game world consists of an infinite number of levels, so called rooms. A room can be best described as an ordinary apartment filled with extraordinary content. A player may interact with the content of the room and then exit through one of the exit doors. Upon leaving a room the player enters a new room. The old room is discarded and cannot be returned to. The new room is once again either generated or loaded from the pool of user-created rooms. This process continues until either the health bar of the player deplete to zero or the player resigns. The objective of the game is to survive as long as possible.

## *1.3 Gameplay*

The basic game mechanics are simple and intuitive - the player controls his avatar using mouse and keyboard. The avatar can interact with the environment, that is pick up items, and fight monsters. A gun, flashlight, and other light sources are the players primary weapons and aids in the game. Light and shadows are a crucial part of the gameplay. E.g. certain monsters can only be damaged through the use of light and are invisible in the shadow.

The avatar has a health bar. The health bar can deplete if the avatar is attacked by monsters. In the case it depletes to zero the avatar dies. Players win or succeed based on survival time.

*1.4 Level Generation*

As already mentioned before, rooms are generated based on chance. This is realized by the room generation algorithm which assigns random values to a set of weighted variables. For instance, for each room there is a variable that dictates whether there is a container with items in it or not. Another variable determines what items (if any) are included in said container. Additionally, whenever a player walks through a door into a new room, there is a chance that he/she will enter a room that another player has created. Due to the random aspect of room generation there is an almost unlimited amount of unique rooms that can be created without the need to design each one individually.

## *1.5 Graphics and Controls*

As mentioned above the game will feature a 2D top down view. We are using a tile-based representation of the game world. For now the tiles are 16x16 pixels and are enlarged to give the game the pixelly, retro look we intended for it. This makes it easy to create models for in-game entities, such as the player, monsters, items and placeables. If users want to use the various editors and add their own models the art won’t be too involved and even inexperienced users will be able to produce satisfactory models given just a little practice. The below image is an in-game screenshot.

 

Figure 1 - In-game screenshot

This conveys the overall graphical feel of the final product. You can see multiple in-game entities on the screen: the player, various passive placeables, items(health hearts & battery), and even two types of monsters.

Also visible in the above screenshot is the dynamic light system. As you can see, the player has a flash light. Light from the flashlight is affected by the architecture & placement of the beam.

Every level will have multiple light sources and then light intensity and angle of radiance will depend on the type of the light source emitting the light. The player’s flash light for example only illuminates a cone-like area and its radius is far smaller than that of a stationary lamp.

## *1.6 Editor*

Our room editor allows users to create & submit their own rooms. For example, if a user creates a room, they will be able to fully customize its architecture and furniture.

## *1.7 Website*

The Website is implemented using a Model-View-Controller because of its ease of development and maintainability.The Website will include the room editor to create and upload custom content to the game. The Website will have to provide consistent experience for the user, it will show the current users information and statistics. In order to register it is as simple as clicking the register button, which would take them to the registration page, and filling in the appropriate information including, login name, email address, and password. Items such as the login name and email address being provided by the potential user or guest would be cross checked by the MVC framework to ensure valid information. This page also has the capability of registered users to login incase they clicked the wrong button for ease of access. As an ease of access to some users, the game and the website would have some overlapping functionalities to make it easier for the users to perform some basic tasks such as creating an account. For this to be possible, there is a web API which the website would call up the appropriate functions. For instance, the URL /webapi/create\_account would be calling a function which both the game and this website would be going through so you could expect the same functionalities.

Once a registered user logs into the game they will see more features including statistics and their custom generated content. The registered users would be able to go to their profile page and change any information that they had originally put down. This includes passwords, email, username, etc.

# **2 Implementation & Changes**

## *2.1 Game Implementation*

 The ROOMS game is implemented in the Haxe programming language. Despite being largely unknown, we chose Haxe because it offered us the following benefit we believed was worth the risk of using a less popular implementation language: Haxe is able to support all of the targets we wish to support (Web, iOS, Android). We initially planned to use Haxe for compiling to C++ on mobile platforms, and Flash on the web. Unfortunately we did not have enough time to implement mobile versions of Rooms because, unknown to us when we chose Haxe, it would have required a significant amount of platform specific code. Unfortunately, after having lost one of our team members, we did not have enough time to do the mobile versions.

 Worth noting is the fact that we did not use a premade game engine, instead, we coded an engine from scratch to support the game mechanics we needed. The reason for this is because we were unable to find a game engine which was sufficiently flexible for our needs (user created content requiring a great deal of flexibility) This took up much of our time, but allowed us to sidestep the time which would normally be spent learning another engine’s API, and the overhead of features which we do not need.

Additionally, all game art was created specifically for use with rooms by team members. Photoshop and Pixen were used primarily, with one team member contributing artwork using Graphics Gale.

 The ROOMS Editor exists only as a level editor currently. It is also implemented in Haxe, and is bundled with the game as a menu option.

## *2.2 Website Implementation*

 The website currently runs on a LAMP stack. LAMP stands for Linux-Apache-MySQL-PHP, which are the technologies behind the web server. We decided to run the website on a linux machine as opposed to a windows machine since it is easier to virtualize, and has more stability for a server in general. Our development environment consists of a virtual machine running stock Debian Squeeze and Apache2.2. There is no desktop environment install on it, so it uses less memory overall. After setting up the virtual machine, we installed MySQL with PHP from the package manager and set up the working directories to pull from our source code repository. This way, the most up-to-date version of the website would be on the demo server because it would update itself everything a commit was sent.

 As mentioned in other sections, we deployed CodeIgniter, which is a MVC framework, for rapid deployment of our website. Most of the basic functions were done so that a presentation could be done with all of our website’s features at least mocked up. The website’s server side code was using PHP, but most of the content displayed was done in HTML and CSS.

 The WebAPI itself was implemented alongside the website. The idea was to design model and interfaces which provided generic enough functions so that we could create a library which can be shared between website functions and Web API functions.

 The demonstration itself was deployed on a personal laptop so that we could have it with us during the demo process in case of failure. Since we used a virtual machine, it was easy to just copy it over and run the virtualization program. Since we were using Git for source code control, it was easy to grab last minute changes from our team members and to deploy it live without having to worry about breaking any functionality with the demonstration itself.

## *2.3 Specification Changes*

 Due to time constraints and the loss of one of our team members, we have had several major changes to our initial specifications. Most notably is the loss of multiplayer functionality and additional content editors. We initially planned for rooms to be a massively multiplayer game, and for there to additionally be item and monster editors. Unfortunately, for the aforementioned reasons, we were unable to finish these features, although we do plan to add them in the future. Finally, as mentioned before, is the loss of a mobile version. We were unable to implement for a mobile platform due to unforeseen challenges with Haxe.

 Despite these changes, we are very happy to report that the rest of our initially proposed feature set has been successfully implemented.