Team D

 Proposal Statement: Team D will design and implement a multiplayer real-time game which is playable in browsers and potentially on mobile devices called *Rooms*. The game features user submitted content. Along with it we will provide a web-based editor that lets players create this content.

 Game concept/ Game play: Rooms is a top-down 2D action, “looting” game. Its graphics and controls are reminiscent of old school Gameboy classics such as the Legend of Zelda. However two key features in combination set it apart from both oldies and modern games - online multiplayer functionality, and the way in which the player affects the game world.

The game world consists of an infinite number of levels, so called rooms. A room always contains four doors and is filled with other content - items, monsters, traps or other players. This content is determined partly by chance and partly by the player’s current stats - time spent in game, past actions, items in inventory. A player may interact with the content of the room and then exit through one of the four doors. Upon leaving a room the player enters a new room. The old room is discarded and cannot be returned to. The new room once again is generated based on chance and the player’s stats. This process continues until either the health bar or hunger bar of the player deplete to zero or the player resigns. The objective of the game is to survive as long as possible.

Now for the most interesting part of the game - the way in which the player affects the game world. As already mentioned before each room is generated based on chance and the player’s stats. This is realized by an algorithm that assigns random values to a set of weighted variables. E.g. 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. Even though the assigned value of the variable is random its weight is entirely determined by the player. This is done in three ways*.*

1) Artifacts. Artifacts are found throughout the game world. They are items that are carried in the player’s inventory but cannot be used actively. Their sole purpose is to affect the various weights that determine the contents of a room. An artifact for example might increase the chances of finding supplies or on the other hand increase the chances of encountering monsters. Players only have a limited amount of inventory slots so they have to chose which artifacts they want to keep and thus in which way they affect the generation of rooms. The effects of artifacts are initially unknown. The player can either determine the effects by carefully observing the changes in the environment or reveal the effects at a level up (more on level ups later).

2) Various player actions. E.g. killing a certain type of monster will increase the chances of encountering that type of monster again.

3) Difficulty - Time passing. The longer the player survives the harder and at the same time more rewarding the game gets. Monsters and items are encountered more frequently. The monsters grow more powerful and the artifacts more useful. If a player has survived for a certain block of time a level up occurs.

At level up the player can chose from five options: reveal stats of an artifact, add an inventory slot, increase health, increase hunger, get a random fact about the game world.

In this way the player learns more and more about the game world and its secrets and is able to influence it too a greater extent.

In the case of a player’s death the player returns to an empty start room and is assigned a random penalty.

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.

The second major component of the game is the multiplayer feature. Throughout the game, each player will have the potential to encounter other players who are also playing the game on their phone or computer.

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 is currently in (instead of randomly generating a new room). Similarly, while a player is exploring a room, at any time another player may enter it. All player to player interactions will occur in real time. Players will be able to communicate over a room chat, team up to travel between rooms together, trade items, attack each other, and use special team items (items that can only be used by a group of players).

Custom Ingame Content: Finally, there is a chance that a player enters one of the custom rooms submitted by a user. This custom content can be created by players using the editors available together with the game.

Editor: Although the game will be playable from multiple platforms, additional functionality will be offered from within the browser. We plan to include a series of editors that will allow users to create & submit their own rooms, monsters, and items. For example, if a user creates a room, they will be able to select the different items and monsters that they would like to have in it. If a user plans to create a monster they will be able to configure reasonable statistics for him and choose a set of pre-defined characteristics that will coorespond to that monster’s AI or characteristics. Users will be able to upload or draw the images for the things they create. Once these things are created, they will be encounterable. A player will have the same chance of encountering user created content as they would normal content, provided the difficulty levels are appropriate for the player’s level.

 Graphics and Controls: As mentioned above the game will feature a 2D top down view. We will include screen mock ups in our presentation to get across the actual interface of the game. As opposed to the deprecated Gameboy controls the controls in rooms will be a combination of mouse and keyboard input if played in the browser and the touch screen if played on a smart phone. As of now it is hard to give any specifics as we will have to experiment with the controls to find the most intuitive and effective way of navigating and interacting with the game world.

Game scope: The entertainment factor of the game greatly depends on the scope of the game content. If there aren’t enough different items and monsters the game will soon lose its draw as the main incentive of playing the game is exploration. It is hard to estimate the exact number of items, monsters and other content we will include in the game. For now we plan on having 20 different archetypes of monsters (random variations of these will be generated by the game itself), each with their own AI behavior, and around 100 custom made items, including various weapons and supply items as well as artifacts. More random artifacts will be generated by the game itself. Since each item will have its own icon including more custom made items does not seem feasible for now. If the game should get enough fan support it would be easy to add more in-game content later on.

Implementation: We will be using a server-client architecture. Our goal is to be able to deploy the client on many platforms, such as web browser or on your mobile phone. Currently, our plan is to use Haxe ([http://www.haxe.org](http://www.haxe.org/)), a language which can compile and run on Android, iOS (via C++ and the NME ([http://www.haxenme.org/](http://www.haxenme.org/%29)) libraries), and the web browser (via flash), however, we’re also investigating other possibilities. For example, an alternative approach would be unity. This would enable us to create 3D models and advanced animations but still keep the 2D controls and top down view. The use of an advanced editor would simplify the development process. The drawback however would be the difficulty of creating 3D meshes for user created content (among other).

The codebase will be maintained by git, and will be built and deployed on a local machine or virtual machine. We will be using MySQL to manage the database containing the user account data. The server-side code will be written in Python, and as the site for the editor needs to interface closely with the server, it will be implemented using Django as a web framework.

 Design Plan: We will use interface based design to implement the server and client. Once the fundamental gameplay is worked out, we will design an API for the server which supports all the basic features we wish to implement. This will allow our team to work on each component (server, client, graphics) separately and in parallel.