.NET Codecamp 2013

Game of Survival

Group 1
Jukka Stranden
Petteri Pekonen

Introduction

The main idea of the Game of Darkness was to implement a board game style computer game for Windows that would use the Windows 8 UI style and which could be expanded easily. So the components would have to be modular in nature.

Idea

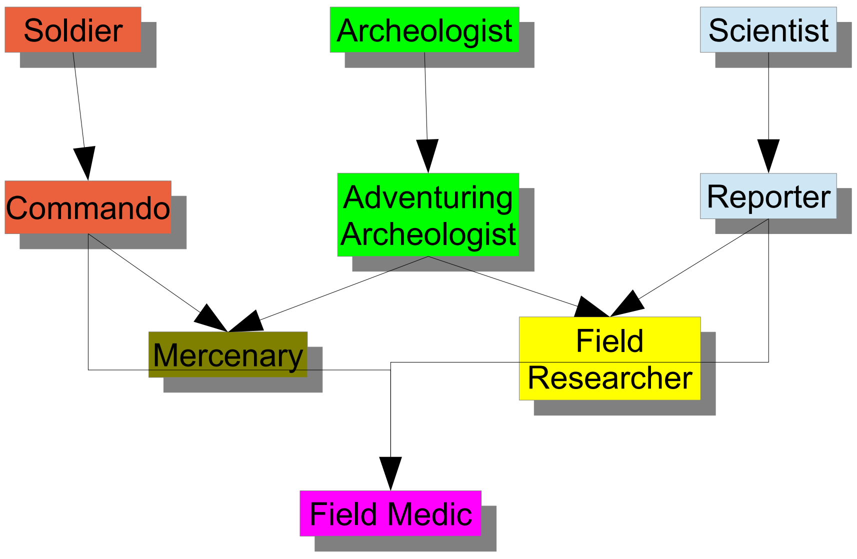
The main idea for the rules and the gamecome from two boardgames: Mansion of Madness (more at: http://www.boardgamegeek.com/boardgame/83330/mansions-of-madness) and Lord of the Rings LCG (more at: http://www.boardgamegeek.com/boardgame/77423/the-lord-of-the-rings-the-card-game). Basically the player is an adventurer who starts at a tomb, ruined mansion etc. She has found whatever she came there to look for and now must escape the place alive. Every turn she gets two moves and an action. She can use these in any order during her turn and the action can be spent to gain additional move. After the player has moved, the game creates a random event for that turn. The event can be a trap that tests one of the player's stats and depending on the result, gives punishment, reward or nothing. The event can also be a new monster that spawns in the player's room. After the event, the monsters move and attack.

The player has three stats and health. The stats are strength, intelligence and dexterity. The test is done by generating a random number between 1 and 10. If the generated number is lower than the player's stat, the player passes the test. Otherwise he fails it. On 1 the player always succeeds and on 10 the player always fails.

The starting stats depend on the player class. Originally we designed the game to have three classes: Soldier with strength and health advantage but intelligence penalty, archeologist with dexterity advantage and average other stats and scientist with intelligence and dexterity but not much health and strength. The idea with the traps was that strength test trap would take health away on fail, dexterity trap would lower a random stat and intelligence trap (a puzzle box) would give a random item as a reward. This way, soldier would be least likely to lose health and strength and could move on, archeologist wouldn't lose stats as easily and would remain competent and scientist would get items that would help her survive. However, we didn't have time to implement the items and classes besides archeologist.

Commercial potential

From the start of the design, the idea was to develop something that could have commercial potential. Ideally a single game would take the player from five to ten minutes and she would earn points in with each run as well as experience for her character. The points would earn her ranking place and the experience could give small advantages to her character, like the ability to negate one event during a game. If a character dies, then that character is permanently loss but the the player could salvage one item or something for her other characters. The game would be free to play but would have in-app purchasing option to buy different things. For example, the player could buy a voodoo spell to resurrect her fallen character. Other things would be to buy small boons like the ability to negate events or protect against stat changes. Each playthrough would also give points towards unlocking more advanced character classes. Or the player could just use a small amount of money to unlock them. The game could have more stats in order to make the variations seem more different. And in addition to that, each character class could have a special power that could be used once during a game. And more of these could be unlocked too to further customize the player character. Classes could even advance in trees. Drawing 1 shows an example on how this could work. After playing a certain amount with one class, the player would unlock a variation of it. After playing enough with that he would unlock half of a hybrid class and after unlocking enough with other class, he would finally unlock the hybrid character.



Drawing 1: Class advancement example

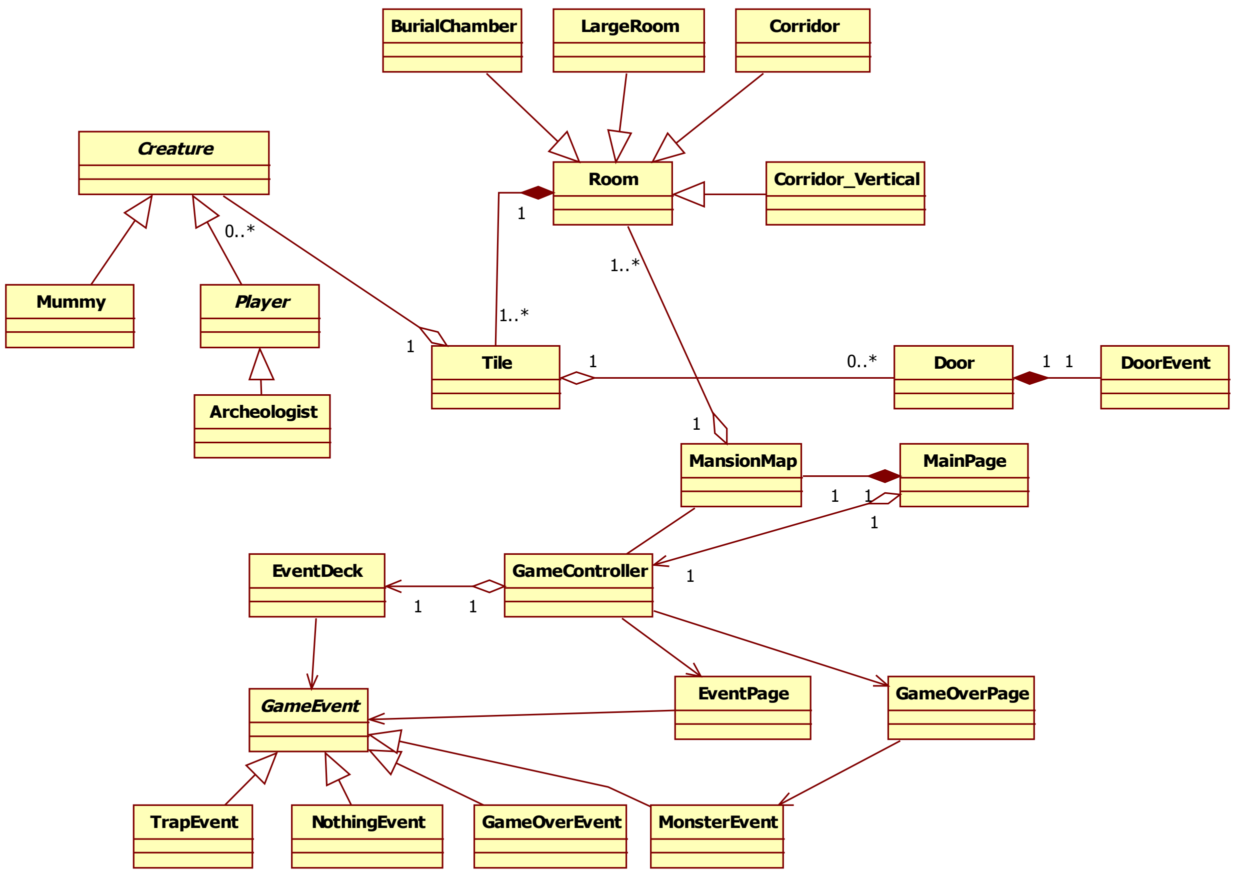
Functionality

The game we were able to finish in the codecamp is a barebone version of the one we planned. Although we knew that we couldn't finish all the features during the codecamp at the beginning, so we set to only implement enough to present the main idea.

The features of this demo version are four custom rooms, archeologist player character, mummy type monster, doors, jammed doors and simple traps. The player can move and attack. Other actions are not available. The game randomly generates one of the three types of traps or a monster at the end of each turn. If the player is in the same tile as a monster, the monster will attack dealing 1 point of damage. The monsters move 1 tile in random direction. After the player has moved to the finish tile (looks like a room with wooden room) and ends turn there, the game ends in player victory. If the player dies, she has lost the game. The game also has a live tile and it updates it at the end of every turn. Player health, score and turn is shown in the live tile.

Technical Design

We created a modular design and had to use lots of inheritance and abstract classes in order to do that. The simplest example of this is Creature class from which both Mummy and Player classes are inherited. The Player class is also abstract and Archeologist (the actual player class) is inherited from the Player class. This way we can deal with players and monsters the same when it comes to tiles and doors. Drawing 2 shows a simplified class diagram of the game. It has just the classes drawn in it.



Drawing 2: Simple class diagram

The game can be broken into two rough pieces: UI and game engine. Between these two is the GameController which enforces the game rules but is also a member of MainPage. MainPage acts as an interface between the GameController and MansionMap.

The game engine consists of the map components: Room, Tile and Door and their inherited classes. Room and Door have graphical presentations but only the URI for the texture, top and left positioning is saved in the classes. Room and Tile object also have the graphical dimensions. Because tile can be of abstract size, its dimensions are required for the creatures to be drawn properly. The classes that are inherited from Room class house the settings for the graphical dimensions and so creating new BurialChamber for example can be done with a single line of code:

```
Room burialChamber = new BurialChamber(300,300);
```

This creates a room object in 300, 300 location on the MansionMap canvas. Well, the Room has to be added to the MansionMap canvas with:

```
map.AddRoom(burialChamber);
```

Doors can be added in similar fashion. Although Doors require the Tiles that the Door connects to. And the Tiles require a reference to the Door because when a tile is clicked, it is checked if the player is next to it, and if it is (and the player has moves left), the player is moved there. If the click is outside the Room but there's a Door, the game attempts to move the player through the Door. The DoorEvent triggers at this moment and depending on the Door, it determines if player can pass through normally or not.

All of this is tied to the player movement. After the player has moved, the GameController asks EventDeck to generate a random event. The Event Deck has a certain percentage for events and depending what the roll is, returns either a MonsterEvent, TrapEvent or NothingEvent. All of the events house the description texts for the UI. The event is processed and then thrown to the EventPage which shows the event description and result.

The bad side of the project is the UI which is currently very static. Everything has an absolute place and size. This would be ok for the map area but the map itself would have to be able to be scrolled. The UI is the thing where most work is required in order to make the application worthy to be applied for the market. New features can be added to the engine with relative ease and it is possible to store map data in database or xml files with ease.

Codecamp

This was not the first Codecamp for either of us and so the work methods as well as knowledge of the time restraints were known to us before the codecamp. We got the application idea from the preceding Saturday when we played a game of Mansions of Madness and thought that it could be implemented during a Codecamp in limited fashion. In retrospect, even that goal was a bit ambitious because writing the engine code took such a long time. We had to work long hours to get the game ready before Friday morning. On Tuesday we quit around 16:20 because another course required the classroom for exercises. On Wednesday we worked 'til 19:00 and had to quit there because we had other events to attend to. On Thursday we started around 9 in the morning and stopped at 1:30 on Thursday-Friday night. Perhaps as a testament to the motivating nature of the codecamp, we weren't even the last group to leave.

At the Codecamp we worked so that Petteri used one of the classroom's computers and Jukka worked with his laptop. We used the Git version control hosted at bitbucket.com to share our code. The version control was essential in quickly merging the code. The work was divided so that Jukka worked with the engine code at first and Petteri worked with the UI and handling the mouse clicks. How to tie the UI to the engine required planning and we had to do a major refactoring at some point. However, in general we worked well and changed roles few times during the codecamp so that one of us developed the UI while the other worked with the engine. We tried to limit conflicts by not working with the same class at the same time but this was always not possible.

We did make some architectural mistakes and had to refactor the whole room system at one point where we tried to make every room its own user control. After that we moved the functionality to MansionMap. However, the UI class still houses many of the game assets like Room and Door lists that should in GameController. Also, we could have used multiple Canvases on top of each other to get more layers and thus avoid some redraws.

Conclusion

The idea statement already describes the greater aspirations for this game. Next steps after the Codecamp would be to first tune the UI so that it would use proper layouts everywhere and while everything inside the MansionMap component could be hardcoded the MansionMap control itself should be scrollable. This would allow us to create larger maps. The next steps after that would be to add more classes, more monsters and events and create a high score list that is listed in a server somewhere. At this point the application would be ready for the Imagine Cup competition. More features such as class advancement and character development could be added after that.

Applying for Microsoft App Campus is also a possibility but the App Campus only accepts mobile apps so we would have to convert the game for WP7 or WP8. For this conversion, the scrollable map area would be even more important.