# -SPECIFICATIONS-

# HELIX’ GAME

**Course Code**: COSC2476/2477

**Course Name**: Game Mechanics and Gameplay Programming

**Assignment Number**: 1

**Marker’s Name**: William Raffe

– Adrian Tang(s3292955)

– Paul Hoang(s3383494)

– Ke Ren(s3405167)

– Isaac Chui(s3379867)

– Paul Poux-Berthe(s3450524)

**Document Description**

|  |  |
| --- | --- |
| **Title** | Specifications of project « Helix »  - Assignment 1 COSC2476/2477 |
| **Creation date** | 17 / 03 / 2014 |
| **Last modification date** | 30 / 03 / 2014 |
| **Author** | Paul Poux-Berthe |
| **In charge** | Paul Poux-Berthe |
| **E-Mail** | [s3450524@student.rmit.edu.au](mailto:s3450524@student.rmit.edu.au) |
| **Key Word** | Game – Procedural Content Generation – C# – Unity 3D |
| **Document Version** | 1.0 |
| **Members** | Adrian Tang(s3292955) – Paul Hoang(s3383494) – Ke Ren(s3405167) – Isaac Chui(s3379867) – Paul Poux-Berthe(s3450524) |

Table of contents

[-SPECIFICATIONS- 1](#_Toc384031827)

[HELIX’ GAME 1](#_Toc384031828)

[Summary 2](#_Toc384031829)

[Presentation of the environment of realization 2](#_Toc384031830)

[Game basic Principles 3](#_Toc384031831)

[Level layout 3](#_Toc384031832)

[Generator level plan 3](#_Toc384031833)

[Magic system 4](#_Toc384031834)

[Player 5](#_Toc384031835)

[Enemies (monsters) 5](#_Toc384031836)

[Team Organization 5](#_Toc384031837)

[Gantt chart 6](#_Toc384031838)

[Sources 7](#_Toc384031839)

## Summary

Helix’ Game is a twelve weeks project as part of RMIT course COSC2476 “Game Mechanics and Game Play Programming”. This project is a team project. The team counts five students. This document is for the developers, the project team and the COSC2476 tutors.

This project is only a desktop application where users can play a game as a single player on different levels through the game story. Furthermore, the game is developed in procedural content generation manner without unbalancing the game mechanics of Helix.

The game is going to be developed with the language C# and the fully integrated development engine Unity 3D.

## Presentation of the environment of realization

In order to realize this game in 12 weeks, the Helix team uses the powerful language C# and its framework. It is a language targeting CLR which is its main advantage, making it easy to integrate with components written in other languages. Moreover, it is a productive and effective language which allows to spend more time on the project than on the code itself. In addition, the C# will be linked to Unity which is a cross-platform game engine with a built-in IDE, a useful tool to create video games. For this game, the team will only be able to use the simple free version of Unity 3D.

## Game basic Principles

Helix’ Game is a top down dungeon crawler with procedural content generation (PCG). Helix is a fast pace, turn based, single player game. A grid layout will be available with isometric view. The objective of the game is to beat the boss at the end to get a new spell with WASD keyboard controls in addition to save and reload the player character. Character death is permanent.

The game will be based on magic colors. Each color will be have a resistance and strength: for example red is strong against pink, but weak against blue. The colors will be a core part of strategic gameplay as both the map and the enemies as well as the spells in the game will be of a certain color. Colors are as follows: red, blue, green, yellow, purple, pink.

## Level layout

The game layout is composed with dungeon crawler elements which are PCG generated. There will be a focus on large open areas with a few corridors or pathways, which will encourage battling against multiple enemies at a time. The map will be built with tiles, each tile holding a player, as well as having a chance to be a certain color. If the player steps on a tile with a color, they will gain the resistances of the color.

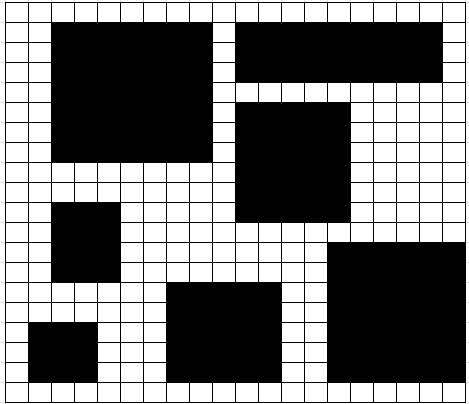
## Generator level plan

1.Create tile with 20X20 grid.

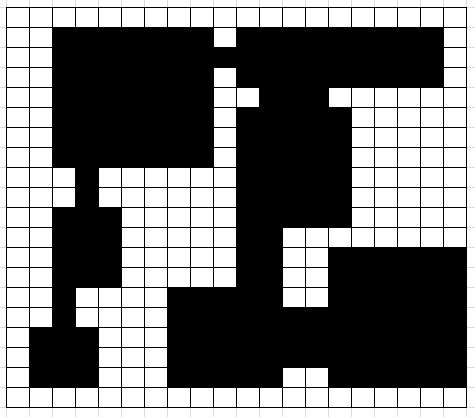
2.Create lists of random land like 1- 5 rooms and each room uses 3X3 to 7X7.

3.Place Room in dungeon, using Tetris algorithmic

4.Using Delaunay triangulation algorithm to connect room to another room without conflict on other path if it already building.

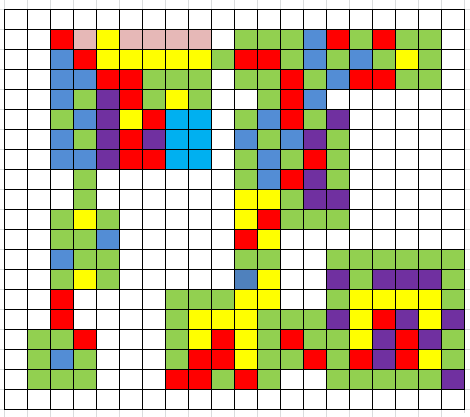


5. Draw tunnel on room to another room with straight way.



6. Create turn based grid system on map.

7.Generator various tile color on passable terrain with grid.



8. First place will be player’s spawn place then other each bigger place will spawn random monster

## Magic system

Helix is a card based game which has the potential of PCG base. The player will cast one card at a time from the top of a deck of cards. Cards have a color and an area which it can affect. These include but are not limited to: Circular, linear, single target, cone, and floor wide targeting. Extras may include shapes such as crosses or Ts. Once the player uses a spell, they will pull out the next card and shuffle the used one back into the deck. We would also like to introduce special effects to the spells, such as knock back, life leech or stuns to showcase the procedural generation more strongly.

## Player

The player will be able to move or attack once per turn. Once the player has done an action, the AI will immediately perform its moves and control is instantly given back to the player (i.e. pseudo-turn based).

The player will have stats that are aligned with the colors (e.g. health is red, magic is blue etc.) These stats can be increased by defeating enemies of the same color. Player’s stats will be a factor when generating new content.

The deck is customizable at the start of each level. There will be a fixed minimum deck size and a variable maximum deck size (perhaps based on the player stats). During gameplay, the player will be able to see the next five or so cards in their deck.

## Enemies (monsters)

There will be several monsters and a single boss on each level. The monsters will be randomly generated on the level and their behavior will be dictated by a state machine. Each monster is given one randomly generated spell. When a monster defeated, stat points are given depending on the color of the enemy.

The boss will be located at the end of the level. The boss will have its own deck of cards, similar to the player. The deck will have a generate-and-test approach and may take into account the collected statistics of the player. When the boss is defeated, a new spell card is given to the player.

Each minion and boss will have basic stats such as health, magic power and armor and may have a color. Both the player and the enemies will do one action per turn (such as move or cast spell).

## Team Organization

To complete Helix’ game and use all the team potential to the success of the project:

- Paul Poux-Berthe: project manager and documentation.

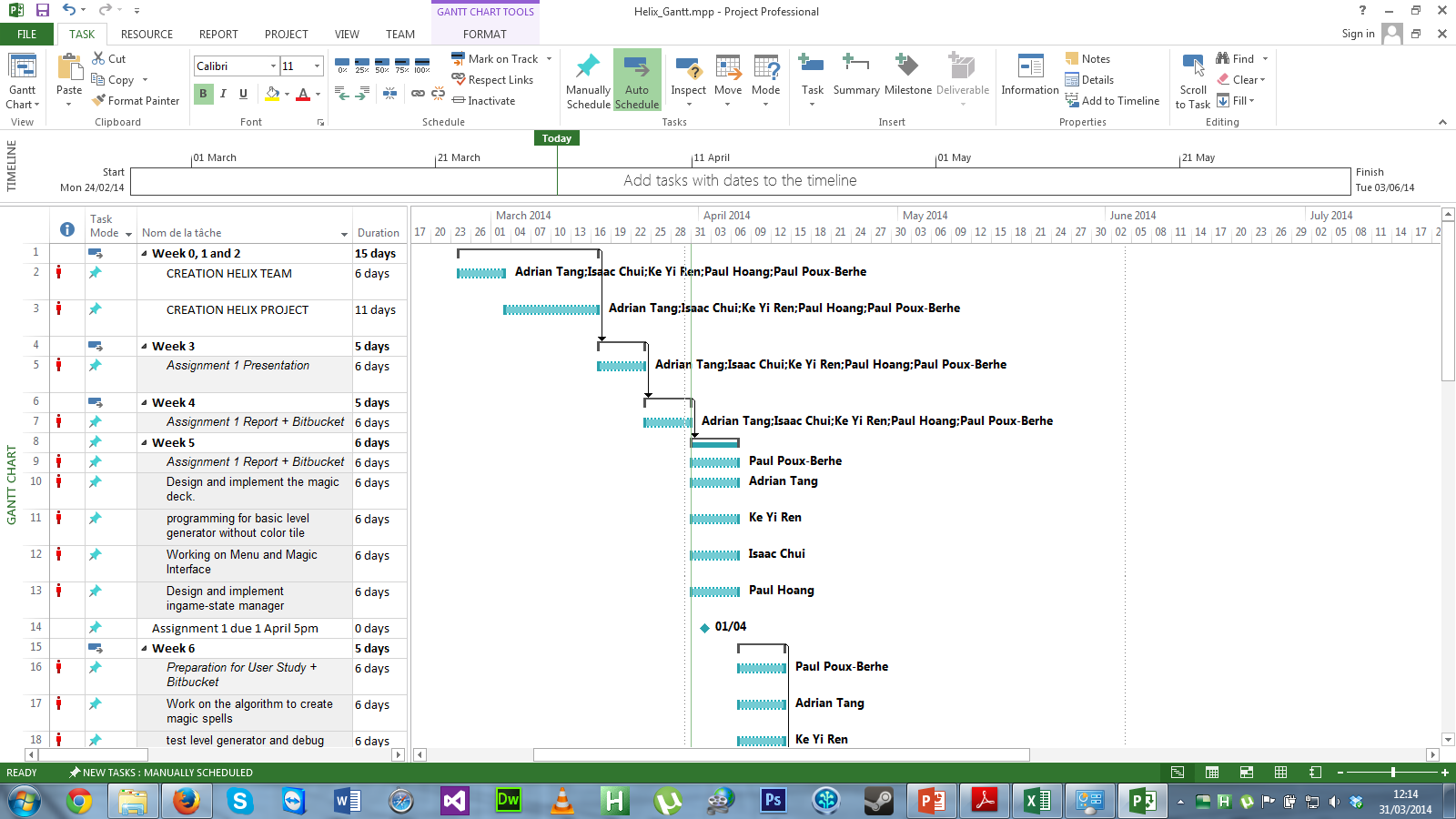
- Adrian Tang: Player scripts and magic system scripts.

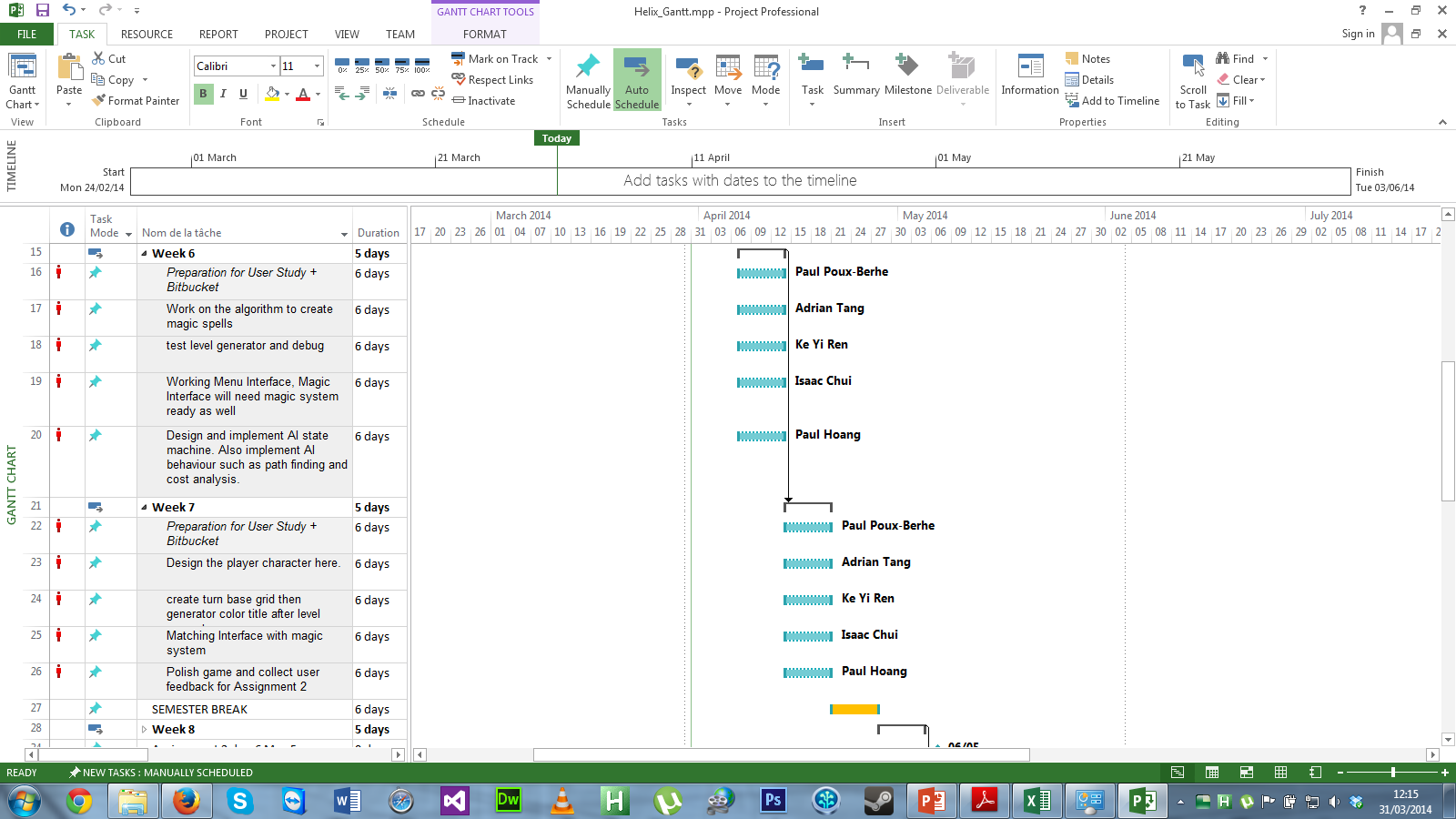
- Paul Hoang: AI manager (C#) + monster + help other if needed.

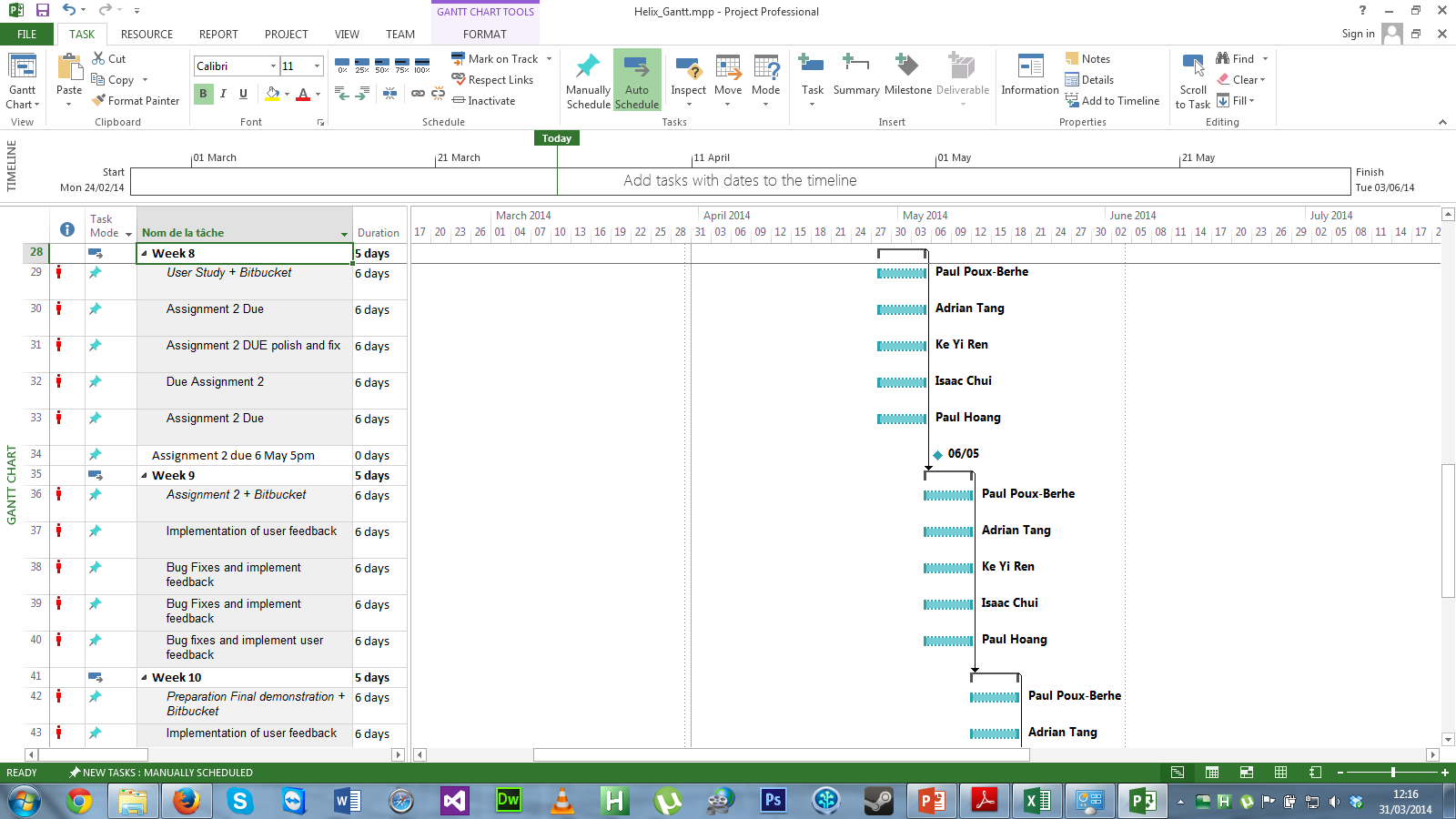
- Ke Yi Ren C#/C++/JavaScript might work on game mechanics, level generator, possible add grid generator for turn based game.

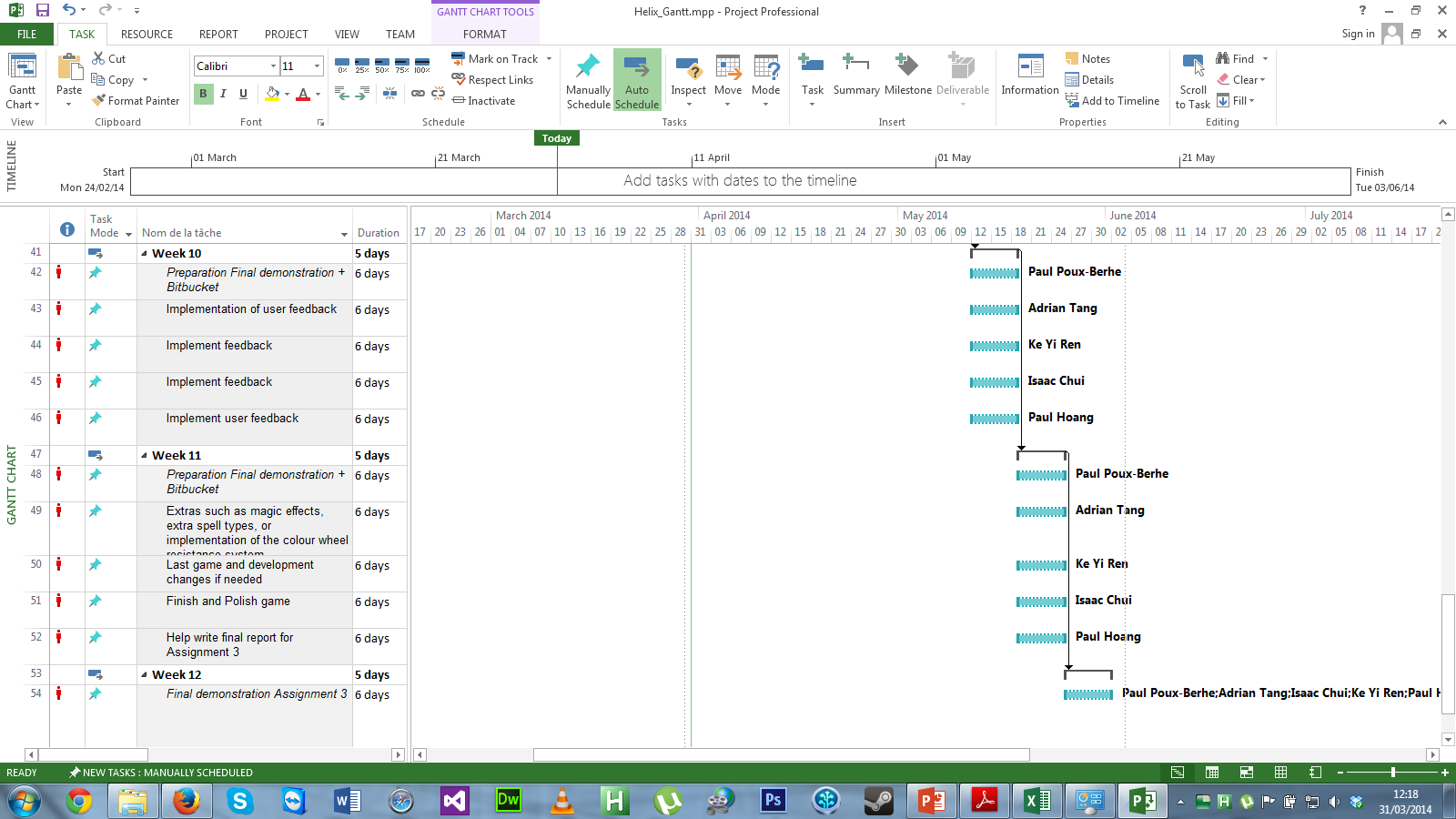
- Isaac Chui: Game manager, menu and GUI manager.

## Gantt chart









## Sources

<http://www.reddit.com/r/gamedev/comments/1dlwc4/procedural_dungeon_generation_algorithm_explained/>

<http://pcg.wikidot.com/pcg-games:angband>

<http://roguelikedeveloper.blogspot.com.au/2007/11/unangband-dungeon-generation-part-two.html>

<http://angband.oook.cz/forum/showthread.php?t=927>

<http://www.roguebasin.com/index.php?title=Basic_BSP_Dungeon_generation>

<http://learninggeekblog.wordpress.com/2013/10/30/dungeon-grind-procedural-dungeon-generation-tutorial/>

<http://dirkkok.wordpress.com/2007/11/21/generating-random-dungeons-part-1/>