Group25MemberContributions.pdf

The classes that **Joseph** Contributed to:

- The Board class
- The Logic class
- The Menu class
- The Player class
- The Movable class
- The "GameTerminalDisplay" class
- "Misc", Miscellaneous Class
- The Game Graphics class
- The File Manager

A description of how Joseph contributed to the implementation of the project:

- I successfully finished the class named Logic, which is dedicated to determining the state of a cell. The menu class was used as the game's main class that the player will navigate in order to play the game. I created the player class, which took in keyboard input, the Movable class, as the abstract parent of all moveable classes. Mathew created the Message of the day class and I overlooked it, which I often did for other peoples classes. With Mathew, I helped write the File Manager.

Other contributions:

- I have been to all of the meetings and was often available for answering other group members calls and text messages, regarding as how to help other individuals with their classes. I have overlooked the implementation of a lot of the game's classes and therefore understood as to what collaborations needed to be had, from other people class implementations.
- With that said, I have contributed to the discussion of the game's implementation.

The unexpected problems during the course of the implementation:

- There was an additional class named 'Misc', for miscellaneous, which was essentially a utility class to be used by other classes. This class primarily dealt with the tile colours such as converting a tile Colour to a java inbuilt colour.
- It was unexpected that I would need to create this class.

The unexpected successes during the course of the implementation:

- The menu class was one of the first classes that I could design, without needing too much interaction between other classes.
- My creation of a smaller board to be used at the start of the implementation week, meant that Eveleen was able to make a swift start on creating the movements for the Floor following thief, the Flying assassin and the Smart thief.

A summary of any significant changes that one made to the design during the implementation phase:

- Instead of creating the menu class with Eveleen, we decided to create the explosion class together. This was done by using some of the code that was within the Flying Assassin class.

The classes that **Eveleen** Contributed to:

- The “FloorFollowingThief” class
- The “SmartThief” class
- The “Explosion” class
- The “FlyingAssassin” class
- The “STTree” class

A description of how Eveleen contributed to the implementation of the project:

- I have implemented the moving direction of the Flying Assassin, The Floor Following Thief, and the Smart Thief. I have also created the “STTree” class, which was intended to check if there were valid movements of direction, which was only to be used for the Smart Thief, because it had to find a path toward the nearest reachable loot or lever

Other contributions:

- I was regularly present during all meetings and consequently I frequently added to the discussion of the games’ implementation.

The unexpected problems during the course of the implementation:

- I didn’t expect to be using a tree data structure for the smart thief. But I felt that this was necessary because the Smart Thief needed to check whether if an Item was found and therefore, the smart Thief could make a path to move toward the direction of an Item.
- The “STTree” class needed to be created, for finding a correct direction.

The unexpected successes during the course of the implementation:

- I did implement the intended classes that were outlined within the design document for A1.
- I found that the creation of the explosion class was relatively straightforward, and the reasoning is outlined below.

A summary of any significant changes that one made to the design during the implementation phase:

- Myself and Joseph decided to create the explosion class together. This was done by using some of the code that was within the Flying Assassin class. This was instead of working on the menu class with Joseph.

The classes that **James** Contributed to:

- The Cell class
- The Enumerated type “TileColour”
- The Misc class
- The Game class

A description of how James contributed to the implementation of the project:

- I implemented the Game class, which contained the board of cell and handled the movement of the bomb, and the items that were impacted by its movement.
- The Cell class was used to store the individual square pieces of the game grid. I also implemented the Cell class to be able to remove the existence of Items, after their collection. This class was also used to set the movable parts of the game, such as the Bomb, and the Cell class was used to check of it’s existence.

Other contributions:

- I have often been present during the meetings and I was available for any group calls that were planned in the evenings. Thus, I often conferred with the team about the game’s implementation.

The unexpected problems during the course of the implementation:

- The Cell and the Game class needed some extra functions.
- I felt that a lot of the functions that were within the UML design, needed to be split into two.

The unexpected successes during the course of the implementation:

- As discussed within the design document, the Cell class was indeed used to store the location of any items. Therefore, the location of any items that were impacted by the detonation of a Bomb.
- As said in the design document, the Cell class was in fact used to return the tile colours, handled by the Logic class.

A summary of any significant changes that one made to the design during the implementation phase:

- For example, within the design document, I stated that The Cell class would be implemented to contain an array of four enumerator colours. Instead, this was handled by other classes and the Cell class contained the individual pieces in the game grid.

The classes that **Mathew** has Contributed to:

- Co-wrote the File Manager
- Message of the day class

Other contributions:

- The "levelfile"
- The creation of the graphics for all of the sprites.
- I have attended all of the meetings and was free for any phone calls that were discussed to be had, during the day. Thus, I have added to the discussion of the game's implementation.

A description of how Mathew contributed to the implementation of the project:

- Collaboratively with Joseph, I co-wrote the File Manager, which is a class that is responsible for converting a game file into format for the game to use.
- I created the Message of the day class. I was in charge of writing the level file format.

The unexpected problems during the course of the implementation:

- I needed to add extra data within the final Level file format, as opposed to what was depicted in the Design document. For example, it was needed to say whether the Smart Thief, Floor Following Thief and the Flying Assassin were going to be moving in an upward or downward direction.

The unexpected successes during the course of the implementation:

- The final implementation of the colours that was within the board, was very similar to the image that I uploaded within the Design document.
- The design document's Level file Format, was used to implement the colours of the board, stayed roughly the same, except for some changes of the colour letters.

A summary of any significant changes that one made to the design during the implementation phase:

- I had to change how the levels were stored in the implementation phase.
- During the design phase, I thought that I should work with Eveleen in order to create the Flying Assassin class. However, it was a better use of my time to focus on the creation of the Level file format along with the graphics. This was because the Thief class was already being implemented by Eveleen.

The classes that **Kamile** Contributed to:

- The Bomb class
- The Clock class
- The Item class
- The Collectible class
- The Gate class
- The Lever class
- The Loot class

Other contributions:

- I has continuously written the weekly minutes for A2, and created the outline for this document.
- I have attended all of the meetings, and therefore often was there to put forward ideas of how I thought that the Items should be implemented.

A description of how Kamile contributed to the implementation of the project:

- I have implemented a class that creates a Bomb that counts down from 3 to detonation, if it has been triggered.
- The Clock class will create a Clock and trigger it's timer. The Gate and Lever class both store the creation of Gates and Lavers, along with their colours.
- The Loot class stores the varying types of Loot and their scores, that will be added to the score from the Game class.
- The classes Collectable and Item are simply parent classes that store the individual types of Items.
- The Bomb and the items, themselves will be stored and fetched from the Cell class.

The unexpected problems during the course of the implementation:

- A lot of time was spent implementing things within the Bomb class, that no longer needed to be implemented because they were being handled within the game class. For example, the Game class handled the bomb's by decrementing the time left on them.

The unexpected successes during the course of the implementation:

- A lot of the design of the CRC cards for the Items were useful within the implementation phase.

A summary of any significant changes that one made to the design during the implementation phase:

- Within the Bomb's CRC card, I outlined that the Bomb class would contain it's self-coordinates. However, it was soon evident that the Game class should handle the moving position of the Bomb, as it also handled other moving NPC's.
- Moreover, the static final tick Rate, outlined within the CRC card for the Bomb, didn't need to be implemented. This was done again by the Game class. This was so that other classes would be able to know that the Bomb class was counting down.