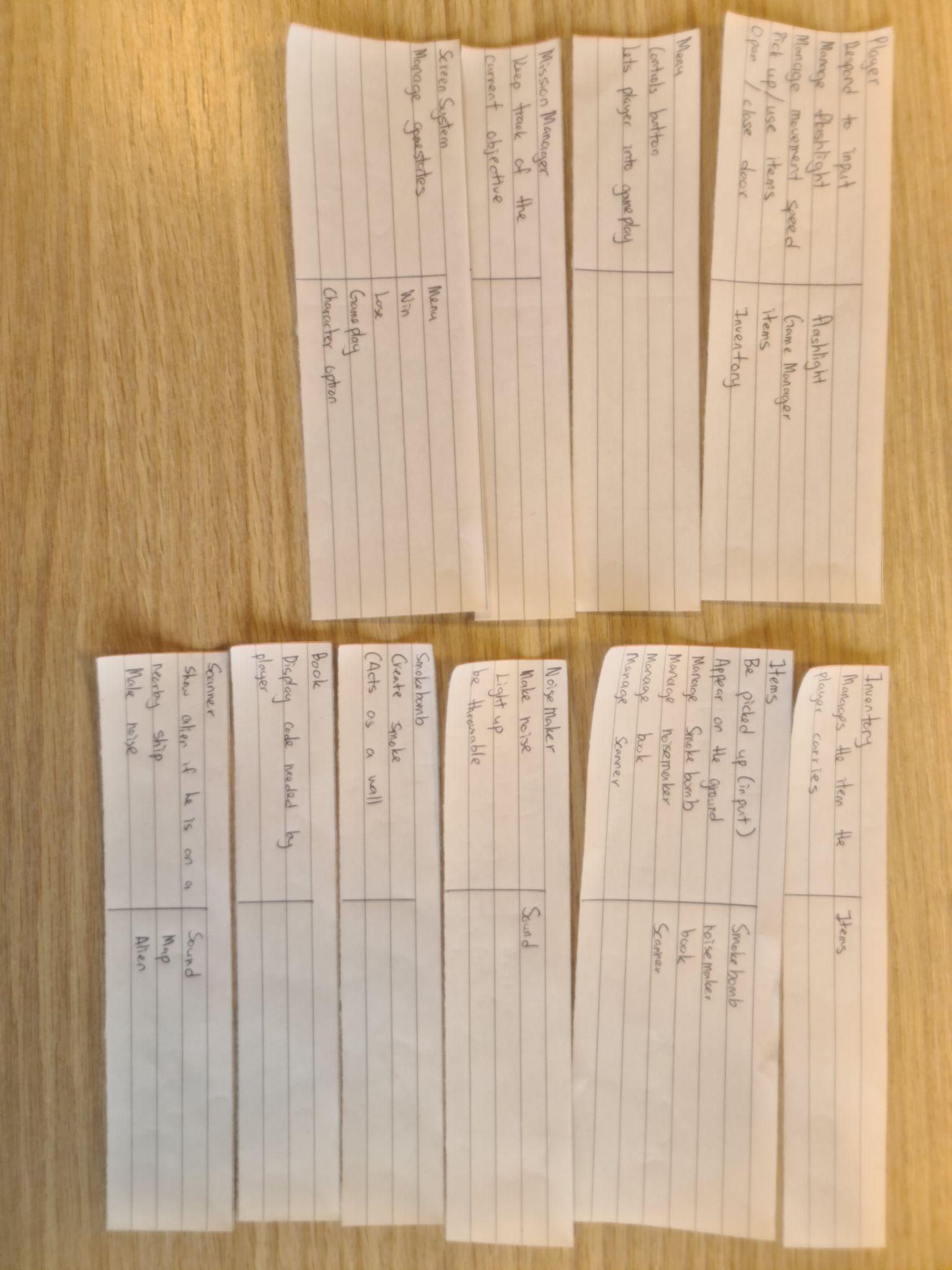
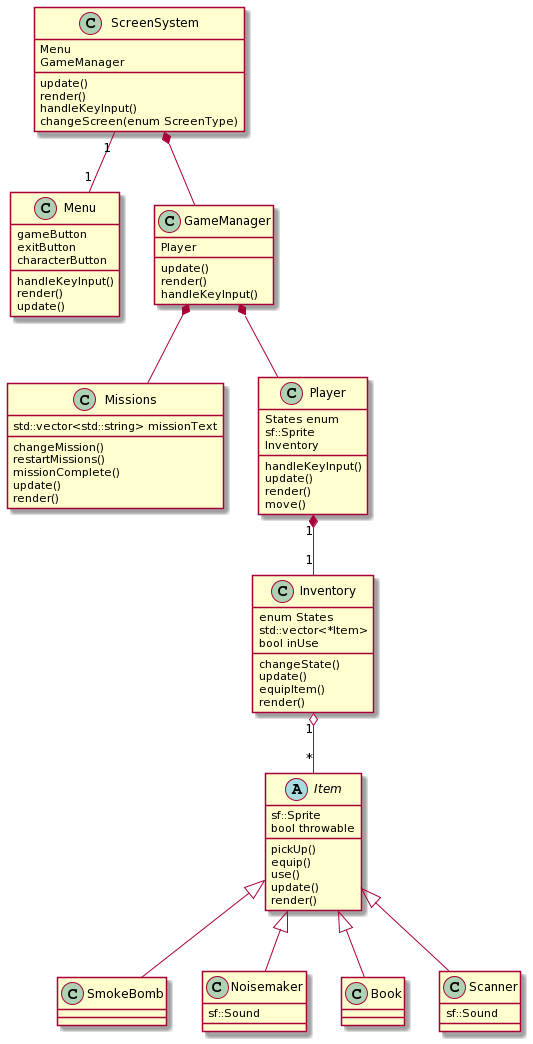
# CRC Cards



# Class Diagram



# Sprint Report 1

## Feature Design

### Feature 1: Player

**Task 1 : Render a rectangle**

Make a rectangle that temporarily represents the player, and display it on the screen.

* void Player::render(sf::RenderWindow)
  + Draw a coloured rectangle on the screen.
  + Have it in the Player class.

**Task 2 : Read player input to move the rectangle**.

Read the WASD keys from the keyboard, and then move the rectangle in the direction corresponding to that key.

* void Player::handleKeyInput()
  + Use the KeyHandler class to check which keys are being pressed, and move the player appropriately.

**Task 3 : Player different movement modes.**

Have an enum class that represents three different states the player will be in - walking, running and sneaking.

* Enum class movementSpeeds  
  {  
   Idle,  
   Sneaking,

Walking,

Running  
   
}

* + Change the player's movement speed, depending on which state the enum is in.

**Task 4: Creating main game loop.**

Set up the game centre loop, and implement the player loop into that.

* void Game::processGameEvents()
  + Player::handleKeyInput()
  + Handle all inputs by the player, and apply them to the right classes
* void Game::update()
  + Player::update(double dt)
  + Handle updating each object in game

### Feature 2: Items

**Task 1: Item parent class**

Make a parent class that will hold most characteristics of the items in the game.

* class Item()  
  {  
   sf::Sprite;  
  }
  + Parent class will save on repeated code, and make the inventory system cleaner.

**Task 2: Make children classes**

Have several smaller classes to make up different kinds of items.

* class smokeBomb : public Item
* class scanner : public Item
* Each child class will differ slightly, for example, the smoke bomb will affect the alien, but the scanner doesn’t.

**Task 3: Picking up items**

Have the player be able to pick up an item if they are facing it, and press the left mouse button.

* Item::getPickedUp()
* Player::handleKeyInput()

**Task 4: Using items**

Have the player be able to use an item after they have picked it up.

* Player::IsKeyPressed() / Player::IsKeyHeldDown
* Item::Trigger();
* Until inventory is done, the player can only carry and use one item at a time.

### Feature 3: Inventory

**Task 1: Store several items**

Have the inventory store the items the player picks up.

* Class Inventory()  
  {  
  std::vector<Item> inventory;  
  }

**Task 2: Make it visible**

Have the player see the inventory when they hit I

* Inventory::handleKeyInput()
* Inventory::render()

**Task 3: Make inventory manager (order, stock)**

Have the inventory organised, and be able to tell the player if they have multiples of different items.

* Inventory::sort()
  + Sort could organise it better for display.

### Feature 4: Grid

**Task 1: Make a square the size of the player, and make an array of those squares.**

* The squares, or cells, will be around the size of the player

**Task 2: Have the cells know which ones will be impassable to the player / alien**

* This is so we can change the movement of those characters depending on the cell

**Task 3: Have the cells know which cells are their neighbours.**

* Good for the A\* pathfinding and level editors.

### Feature 5: Alien

**Task 1: Make the alien be able to move to a point given to it.**

* This will be the start of its patrolling system.

**Task 2: Set up the randomised patrolling points**

* The alien goes to the same 4 points, but once it reaches one point, the next one it goes to will be random.

**Task 3: Make it so the alien has a vision cone.**

* Have it be able to react to when it spots the player in front of them.

**Task 4: Make the alien be able to hear.**

* No sound is implemented yet, so it will just be button presses
* Makes the alien deviate from its original path

## Summary of planned work

|  |  |  |
| --- | --- | --- |
| Feature and tasks | Time(Hours) | Team member |
| **Feature 1: Player** | 6 |  |
| **Task 1 : Render a rectangle** | 1 | Danial |
| **Task 2 : Read player input to move the rectangle**. | 1 | Danial |
| **Task 3 : Player different movement modes.** | 2 | Danial |
| **Task 4: Creating main game loop.** | 2 | Danial |
| **Feature 2: Items** | 9 |  |
| **Task 1: Item parent class** | 3 | Caroline |
| **Task 2: Make children classes** | 2 | Caroline |
| **Task 3: Picking up items** | 1 | Caroline |
| **Task 4: Using items** | 3 | Caroline |
| **Feature 3: Inventory** | 6 |  |
| **Task 1: Store several items** | 3 | Caroline |
| **Task 2: Make it visible** | 1 | Caroline |
| **Task 3: Make inventory manager (order, stock)** | 2 | Caroline |
| **Feature 4: Grid** | 5 |  |
| **Task 1: Make a square the size of the player, and make an array of those squares.** | 1 | Danial |
| **Task 2: Have the cells know which ones will be impassable to the player / alien** | 2 | Danial |
| **Task 3: Have the cells know which cells are their neighbours.** | 2 | Danial |
| **Feature 5: Alien** | 7 |  |
| **Task 1: Make the alien be able to move to a point given to it.** | 2 | Danial |
| **Task 2: Set up the randomised patrolling points** | 1 | Danial |
| **Task 3: Make it so the alien has a vision cone.** | 2 | Danial |
| **Task 4: Make the alien be able to hear** | 2 | Caroline |
| **Total** | 33 |  |

## Work Completed

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature and tasks | Time Estimate | Time Actual | Team member | Complete |
| **Feature 1: Player movement** | 6 | 6.5 |  |  |
| **Task 1 : Render a rectangle** | 1 | 0.5 | Danial | YES |
| **Task 2 : Read player input to move the rectangle**. | 1 | 3 | Danial | YES |
| **Task 3 : Player different movement modes.** | 2 | 1 | Danial | YES |
| **Task 4: Creating main game loop.** | 2 | 2 | Danial | YES |
| **Feature 2: Items** | 9 | 18 |  |  |
| **Task 1: Item parent class** | 3 | 5 | Caroline | Yes |
| **Task 2: Make children classes** | 2 | 8 | Caroline | Yes |
| **Task 3: Picking up items** | 1 | 2 | Caroline | Yes |
| **Task 4: Using items** | 3 | 3 | Caroline | Yes |
| **Feature 3: Inventory** | 6 | 11 |  |  |
| **Task 1: Store several items** | 3 | 5 | Caroline | YES |
| **Task 2: Make it visible** | 1 | 2 | Caroline | YES |
| **Task 3: Make inventory manager (order, stock)** | 2 | 4 | Caroline | YES |
| **Feature 4: Grid** | 5 | 14 |  |  |
| **Task 1: Make a square the size of the player, and make an array of those squares.** | 1 | 3 | Danial | YES |
| **Task 2: Have the cells know which ones will be impassable to the player / alien** | 2 | 7 | Danial | YES |
| **Task 3: Have the cells know which cells are their neighbours.** | 2 | 4 | Danial | YES |
| **Feature 5: Alien** | 7 | 14 |  |  |
| **Task 1: Make the alien be able to move to a point given to it.** | 2 | 4 | Danial | YES |
| **Task 2: Set up the randomised patrolling points** | 1 | 2 | Danial | YES |
| **Task 3: Make it so the alien has a vision cone.** | 2 | 4 | Danial | YES |
| **Task 4: Make the alien be able to hear** | 2 | 4 | Caroline | YES |
|  | 33 |  |  |  |

# Sprint Report 2

### Player Mission

**Task 1: Display mission**

Show the text of the display on the top right (temporarily, it will go into the main map later)

* MissionSystem::render()

**Task 2: Mission order**

Organise the mission text into an order to be displayed to the player one after another.

* MissionSystem::std::array<string> missions;
  + This is also where we decide what the missions will be.

**Task 3: Update mission**

This will detect when to change the text in the top right to a new objective.

* MissionSystem::update(double dt);

**Task 4: Complete all tasks**

Have the code be able to tell when all tasks are complete, and be ready for when the win screen is coded.

* MissionSystem::end()
* For now, display some text saying you win.

### UI ( Menu Only )

**Task 1: Screen system**

Set up the basic screen changing code.

* GameState::Menu;
* This is only for the menu system, the other gamestates will be added later.

**Task 2: Menu layout**

Have a very basic looking menu with buttons you can mouse over.

* sf::Button::OnMouse.Hover()
* Have two buttons, one for the game, and one to exit.

**Task 3: Going to game through buttons**

Being able to click on the buttons, either to close the game or to go to the gameplay.

* sf::Button::OnMouse.Click()