Under Us

A Sea Exploration Adventure
(The Hunt for the Lost Sea Cucumber)

[Tali]

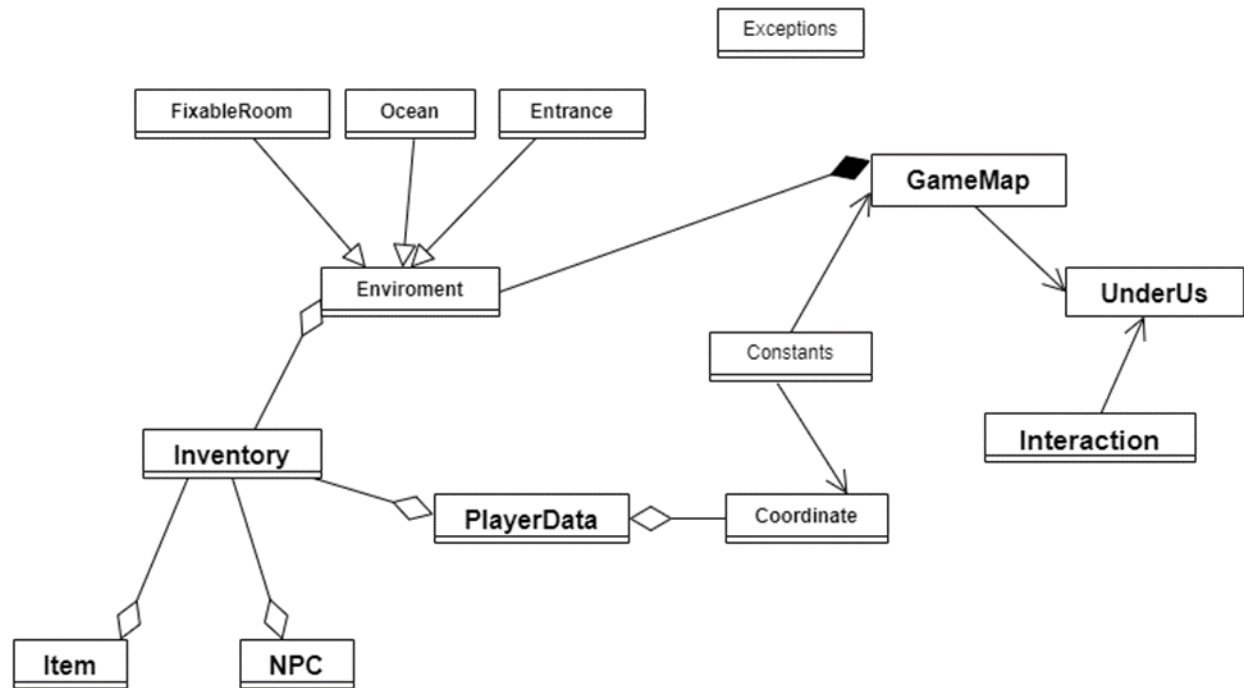**[Team Members:]**

Logan Feit
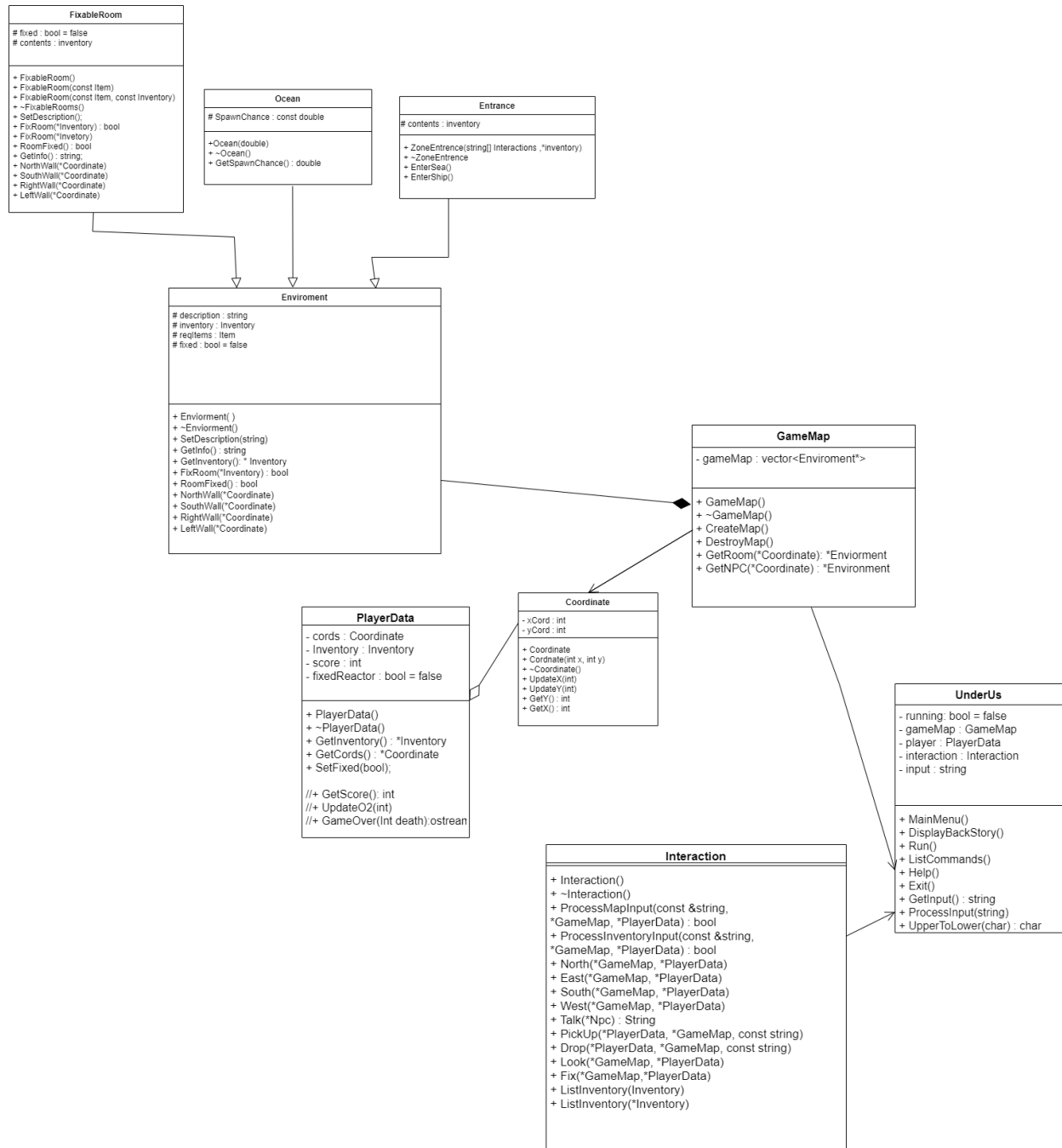
Keagan Rieder

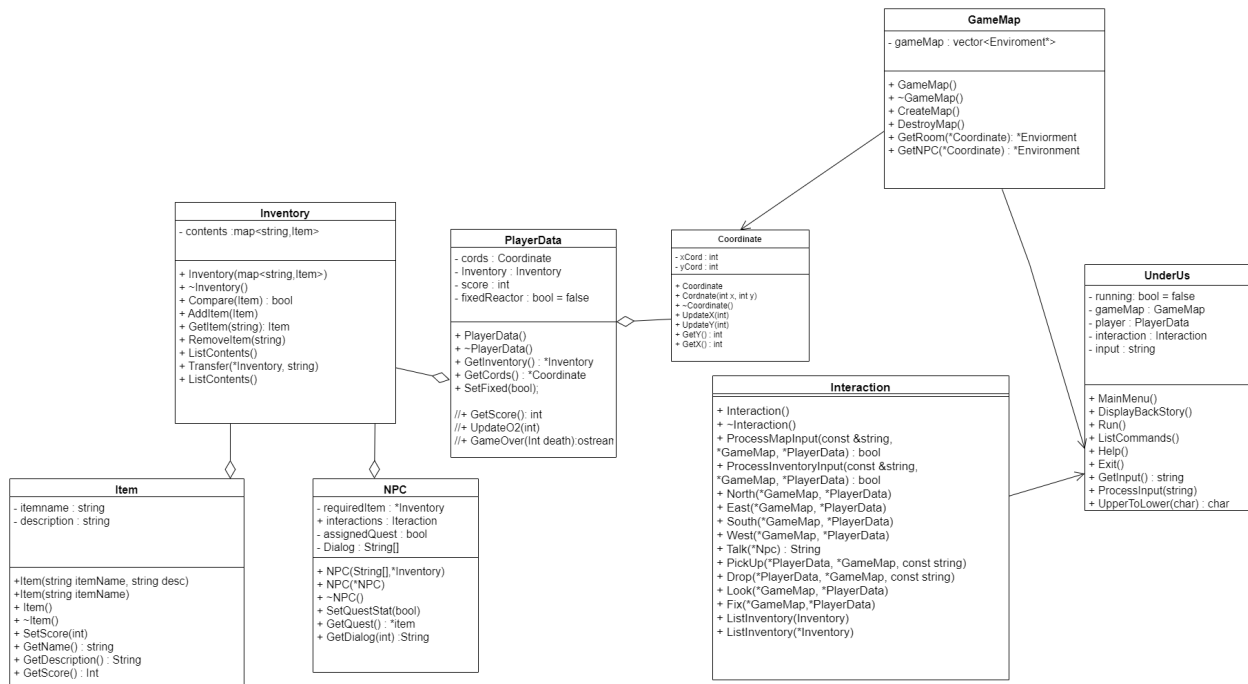**Maxenne Jubane**

Jenil Jani

Software Design

Design – Class diagrams

MapInteraction:

FixableRoom

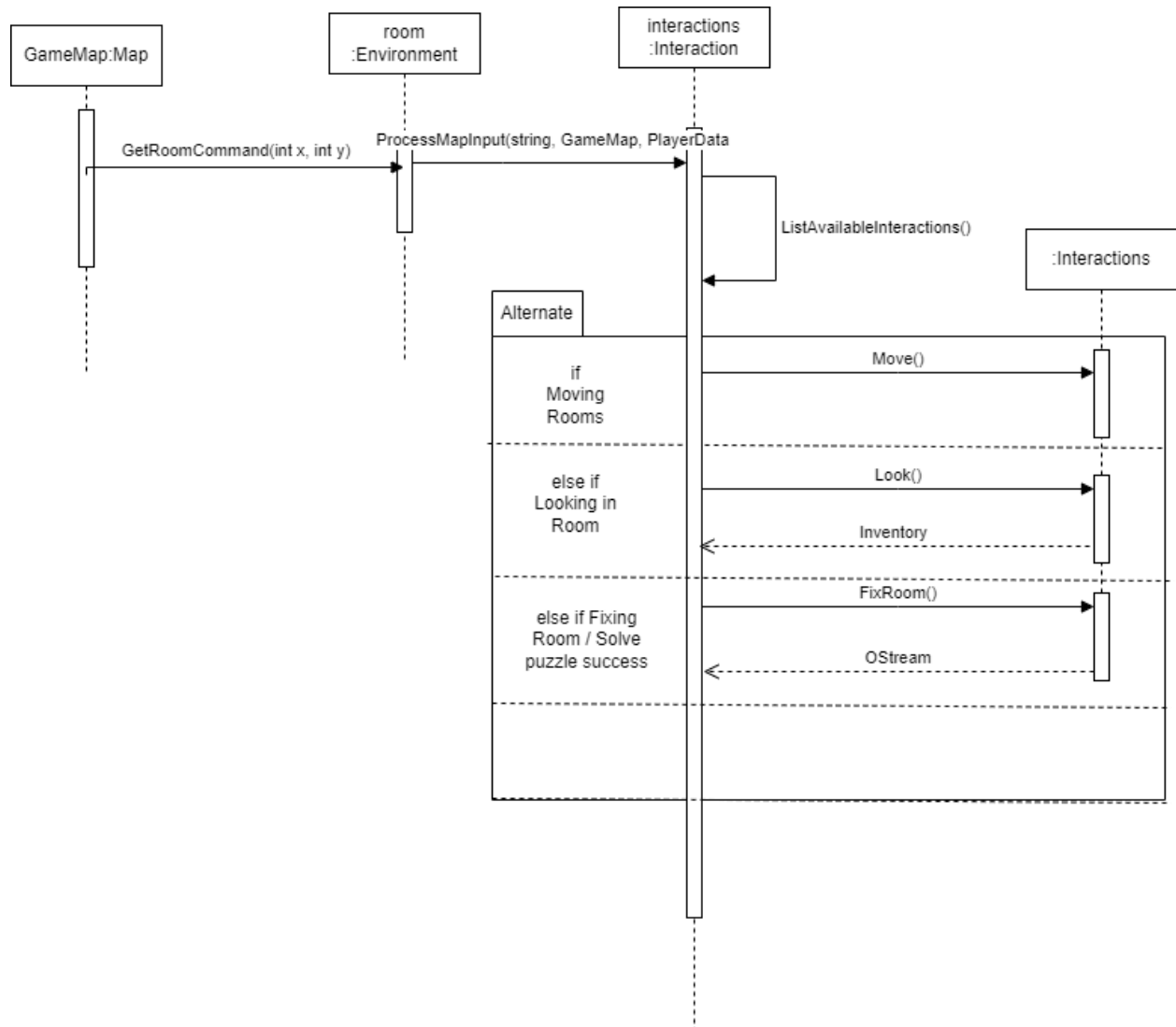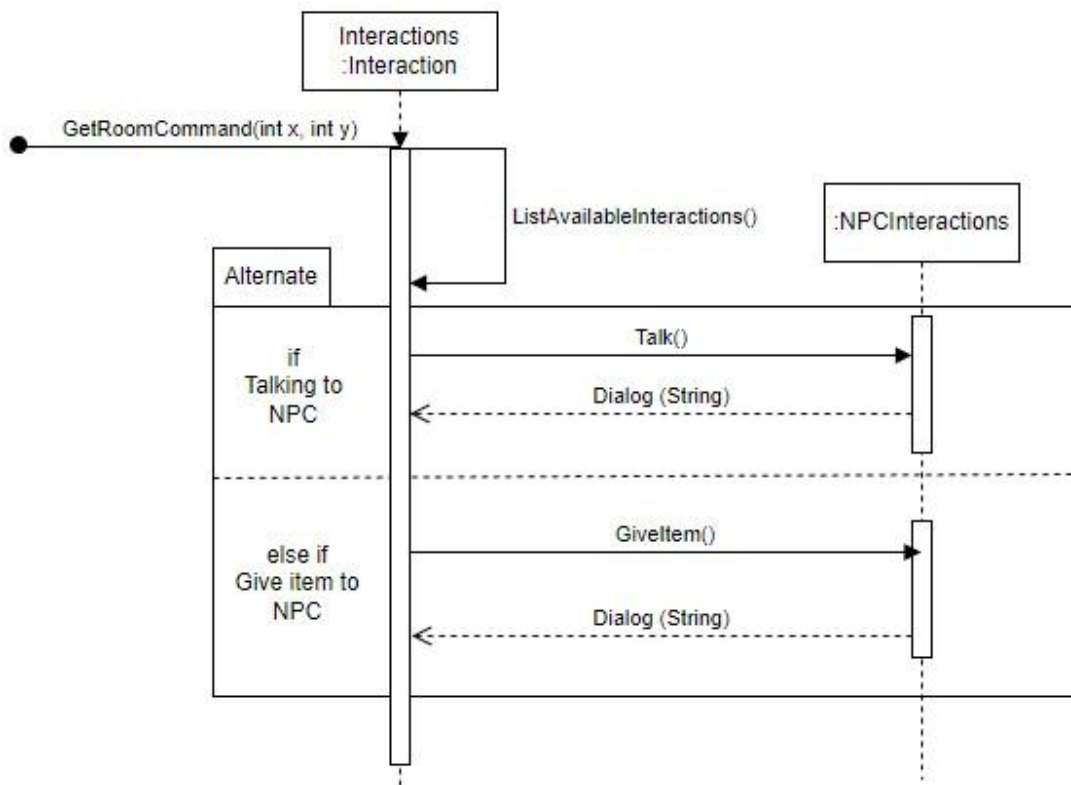\# fixed : bool = false
\# contents : inventory

\+ FixableRoom()
\+ FixableRoom(const Item)
\+ FixableRoom(const Item, const Inventory)
\+ ~FixableRooms()
\+ SetDescription();
\+ FixRoom(*Inventory) : bool
\+ FixRoom(*Inventory)
\+ RoomFixed() : bool
\+ GetInfo() : string;
\+ NorthWall(*Coordinate)
\+ SouthWall(*Coordinate)
\+ RightWall(*Coordinate)
\+ LeftWall(*Coordinate)

Ocean

\# SpawnChance : const double

\+Ocean(double)
\+ ~Ocean()
\+ GetSpawnChance() : double

Entrance

\# contents : inventory

\+ ZoneEntrence(string[] Interactions ,*inventory)
\+ ~ZoneEntrence
\+ EnterSea()
\+ EnterShip()

Enviroment

\# description : string
\# inventory : Inventory
\# reqItems : Item
\# fixed : bool = false

\+ Enviorment( )
\+ ~Enviorment()
\+ SetDescription(string)
\+ GetInfo() : string
\+ GetInventory(): * Inventory
\+ FixRoom(*Inventory) : bool
\+ RoomFixed() : bool
\+ NorthWall(*Coordinate)
\+ SouthWall(*Coordinate)
\+ RightWall(*Coordinate)
\+ LeftWall(*Coordinate)

GameMap

\- gameMap : vector<Enviroment*>

\+ GameMap()
\+ ~GameMap()
\+ CreateMap()
\+ DestroyMap()
\+ GetRoom(*Coordinate) : *Enviorment
\+ GetNPC(*Coordinate) : *Environment

PlayerData

\- cords : Coordinate
\- Inventory : Inventory
\- score : int
\- fixedReactor : bool = false

\+ PlayerData()
\+ ~PlayerData()
\+ GetInventory() : *Inventory
\+ GetCords() : *Coordinate
\+ SetFixed(bool);

//+ GetScore(): int
//+ UpdateO2(int)
//+ GameOver(Int death):ostrean

Coordinate

\- xCord : int
\- yCord : int

\+ Coordinate
\+ Cordnate(int x, int y)
\+ ~Coordinate()
\+ UpdateX(int)
\+ UpdateY(int)
\+ GetY() : int
\+ GetX() : int

UnderUs

\- running: bool = false
\- gameMap : GameMap
\- player : PlayerData
\- interaction : Interaction
\- input : string

\+ MainMenu()
\+ DisplayBackStory()
\+ Run()
\+ ListCommands()
\+ Help()
\+ Exit()
\+ GetInput() : string
\+ ProcessInput(string)
\+ UpperToLower(char) : char

Interaction

\+ Interaction()
\+ ~Interaction()
\+ ProcessMapInput(const &string, *GameMap, *PlayerData) : bool
\+ ProcessInventoryInput(const &string, *GameMap, *PlayerData) : bool
\+ North(*GameMap, *PlayerData)
\+ East(*GameMap, *PlayerData)
\+ South(*GameMap, *PlayerData)
\+ West(*GameMap, *PlayerData)
\+ Talk(*Npc) : String
\+ PickUp(*PlayerData, *GameMap, const string)
\+ Drop(*PlayerData, *GameMap, const string)
\+ Look(*GameMap, *PlayerData)
\+ Fix(*GameMap,*PlayerData)
\+ ListInventory(Inventory)
\+ ListInventory(*Inventory)

Item & NPC Interaction:

GameMap

- gameMap : vector<Enviroment*>

+ GameMap()
+ ~GameMap()
+ CreateMap()
+ DestroyMap()
+ GetRoom(*Coordinate): *Enviorment
+ GetNPC(*Coordinate) : *Environment

Inventory

- contents :map<string,Item>

+ Inventory(map<string,Item>)
+ ~Inventory()
+ Compare(Item) : bool
+ AddItem(Item)
+ GetItem(string): Item
+ RemoveItem(string)
+ ListContents()
+ Transfer(*Inventory, string)
+ ListContents()

PlayerData

- cords : Coordinate
- Inventory : Inventory
- score : int
- fixedReactor : bool = false

+ PlayerData()
+ ~PlayerData()
+ GetInventory() : *Inventory
+ GetCords() : *Coordinate
+ SetFixed(bool);

//+ GetScore(): int
//+ UpdateO2(int)
//+ GameOver(Int death):ostrean

Coordinate

- xCord : int
- yCord : int

+ Coordinate
+ Cordnate(int x, int y)
+ ~Coordinate()
+ UpdateX(int)
+ UpdateY(int)
+ GetY() : int
+ GetX() : int

UnderUs

- running: bool = false
- gameMap : GameMap
- player : PlayerData
- interaction : Interaction
- input : string

+ MainMenu()
+ DisplayBackStory()
+ Run()
+ ListCommands()
+ Help()
+ Exit()
+ GetInput() : string
+ ProcessInput(string)
+ UpperToLower(char) : char

Item

- itemname : string
- description : string

+Item(string itemName, string desc)
+Item(string itemName)
+ Item()
+ ~Item()
+ SetScore(int)
+ GetName() : string
+ GetDescription() : String
+ GetScore() : Int

NPC

- requiredItem : *Inventory
+ interactions : Iteraction
- assignedQuest : bool
- Dialog : String[]

+ NPC(String[],*Inventory)
+ NPC(*NPC)
+ ~NPC()
+ SetQuestStat(bool)
+ GetQuest() : *item
+ GetDialog(int) :String

Interaction

+ Interaction()
+ ~Interaction()
+ ProcessMapInput(const &string, *GameMap, *PlayerData) : bool
+ ProcessInventoryInput(const &string, *GameMap, *PlayerData) : bool
+ North(*GameMap, *PlayerData)
+ East(*GameMap, *PlayerData)
+ South(*GameMap, *PlayerData)
+ West(*GameMap, *PlayerData)
+ Talk(*Npc) : String
+ PickUp(*PlayerData, *GameMap, const string)
+ Drop(*PlayerData, *GameMap, const string)
+ Look(*GameMap, *PlayerData)
+ Fix(*GameMap,*PlayerData)
+ ListInventory(Inventory)
+ ListInventory(*Inventory)

FixableRoom

fixed : bool = false
contents : inventory

+ FixableRoom()
+ FixableRoom(const Item)
+ FixableRoom(const Item, const Inventory)
+ ~FixableRooms()
+ SetDescription();
+ FixRoom(*Inventory) : bool
+ FixRoom(*Inventory)
+ RoomFixed() : bool
+ GetInfo() : string;
+ NorthWall(*Coordinate)
+ SouthWall(*Coordinate)
+ RightWall(*Coordinate)
+ LeftWall(*Coordinate)

Ocean

SpawnChance : const double

+Ocean(double)
+ ~Ocean()
+ GetSpawnChance() : double

Entrance

contents : inventory

+ ZoneEntrance(string[] Interactions ,*inventory)
+ ~ZoneEntrence
+ EnterSea()
+ EnterShip()

Enviroment

description : string
inventory : Inventory
reqItems : Item
fixed : bool = false

+ Enviroment( )
+ ~Enviroment()
+ SetDescription(string)
+ GetInfo() : string
+ GetInventory() : * Inventory
+ FixRoom(*Inventory) : bool
+ RoomFixed() : bool
+ NorthWall(*Coordinate)
+ SouthWall(*Coordinate)
+ RightWall(*Coordinate)
+ LeftWall(*Coordinate)

Exceptions

+Invalid_parameter_error(const char*
errMessege)
~Calculation_error(const char*
errMessege)

GameMap

- gameMap : vector<Enviroment*>

+ GameMap()
+ ~GameMap()
+ CreateMap()
+ DestroyMap()
+ GetRoom(*Coordinate): *Enviroment
+ GetNPC(*Coordinate) : *Environment

Constants

Room : enum {
HallWay,
Reactor
Other }

Inventory

- contents :map<string,Item>

+ Inventory(map<string,Item>)
+ ~Inventory()
+ Compare(Item) : bool
+ AddItem(Item)
+ GetItem(string): Item
+ RemoveItem(string)
+ ListContents()
+ Transfer(*Inventory, string)
+ ListContents()

PlayerData

- cords : Coordinate
- Inventory : Inventory
- score : int
- fixedReactor : bool = false

+ PlayerData()
+ ~PlayerData()
+ GetInventory() : *Inventory
+ GetCords() : *Coordinate
+ SetFixed(bool);

//+ GetScore(): int
//+ UpdateO2(int)
//+ GameOver(Int death):ostrean

Coordinate

- xCord : int
- yCord : int

+ Coordinate
+ Cordnate(int x, int y)
+ ~Coordinate()
+ UpdateX(int)
+ UpdateY(int)
+ GetY() : int
+ GetX() : int

UnderUs

- running: bool = false
- gameMap : GameMap
- player : PlayerData
- interaction : Interaction
- input : string

+ MainMenu()
+ DisplayBackStory()
+ Run()
+ ListCommands()
+ Help()
+ Exit()
+ GetInput() : string
+ ProcessInput(string)
+ UpperToLower(char) : char

Interaction

+ Interaction()
+ ~Interaction()
+ ProcessMapInput(const &string,
*GameMap, *PlayerData) : bool
+ ProcessInventoryInput(const &string,
*GameMap, *PlayerData) : bool
+ North(*GameMap, *PlayerData)
+ East(*GameMap, *PlayerData)
+ South(*GameMap, *PlayerData)
+ West(*GameMap, *PlayerData)
+ Talk(*Npc) : String
+ PickUp(*PlayerData, *GameMap, const string)
+ Drop(*PlayerData, *GameMap, const string)
+ Look(*GameMap, *PlayerData)
+ Fix(*GameMap, *PlayerData)
+ ListInventory(Inventory)
+ ListInventory(*Inventory)

Item

- itemname : string
- description : string

+Item(string itemName, string desc)
+Item(string itemName)
+ Item()
+ ~Item()
+ SetScore(int)
+ GetName() : string
+ GetDescription() : String
+ GetScore() : Int

NPC

- requiredItem : *Inventory
+ interactions : IIeraction
- assignedQuest : bool
- Dialog : String[]

+ NPC(String[],*Inventory)
+ NPC(*NPC)
+ ~NPC()
+ SetQuestStat(bool)
+ GetQuest() : *item
+ GetDialog(int) :String

Design – sequence diagrams

Interactions :Interaction

GetRoomCommand(int x, int y)

ListAvailableInteractions()

:NPCInteractions

Alternate

if
Talking to NPC

Talk()

Dialog (String)

else if
Give item to NPC

GiveItem()

Dialog (String)

```
                    ┌──────────────────┐
                    │   interactions   │
                    │   :Interaction   │
                    └──────────────────┘
                             ┆
                             ┆
  ProcessInventoryInput(string, GameMap, PlayerData)
  ◄──────────────────────────────────┐
                                   ┌──┴──┐
                                   │     │  ListAvailableInteractions()
                                   │     │                              ┌──────────────┐
                                   │     │                              │ :Interactions │
                                   ◄─────┘                              └──────────────┘
     ┌──────────┐                  │                                           ┆
     │ Alternate │                 │                                           ┆
  ┌──┴──────────┴──────────────────┼───────────────────────────────────────────────────┐
  │              if                │         ListInventory()                   ┌─┐       │
  │         Listing item           ├──────────────────────────────────────────► │       │
  │                                │           Inventory                       │ │       │
  │                                ◄ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─┤ │       │
  ├ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┼ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ └─┘─ ─ ─ ─┤
  │                                │      Pickup(player, map, itemName)        ┌─┐       │
  │           else if              ├──────────────────────────────────────────► │       │
  │         Pickup item            │           Inventory : item                │ │       │
  │                                ◄ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─┤ │       │
  ├ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┼ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ └─┘─ ─ ─ ─┤
  │                                │        Drop(player, map, itemName)        ┌─┐       │
  │         else if Drop           ├──────────────────────────────────────────► │       │
  │                                │           Inventory : item                │ │       │
  │                                ◄ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─┤ │       │
  └────────────────────────────────┼──────────────────────────────────────────└─┘───────┘
                                    │
                                    │
```

Class Descriptions

Game Overview

Plot

·　Captain's dead, communications gone, engine is not working. You're a crew member who has to collect items to fix the problems and maybe make some sea cucumber friends along the way.

·　First the Soldier man sends you to look for engine parts, you get it, fix it, talk to the Soldier again.

·　Second the chef tells you about missing food and equipment, he sends you to look for the source. Gives you keys and a flashlight.

·　Looking at the newly unlocked rooms you find the Stowaway who has been causing a ruckus in the submarine.

·　During the interrogation on the Stowaway the comms for the submarine is cut. You must find spare parts underwater to fix this grave emergencies.

·　The underwater section is unlocked and now explorable. You try to find comms pieces in the dangerous waters. What you do and find there from the randomly generated maps will lead to what ending you get.

Environment

·　Allows for the deceleration/creation of rooms and locations inside of the game. There are two classes which inherit from this class and allow the formation of more specialized rooms able to do things the environment can't. These are ocean and fixable rooms and map entrances

Variables
name : string
fixed : bool = false
*inventory : Inventory
*requiredItems : Inventory
*npc : Npc = nullptr
Member functions:
virtual SetContents () = 0
+ Environment ()
+ ~Environment ()
+ GetType () : string
+ virtual GetInfo () = 0
+virtual GetInventory () : Inventory*
+ virtual GetReqInventory () : Inventory*
+virtual GetNPC() : Npc*
+ virtual Fixroom (*Inventory)
+ virtual RoomFixed () : bool

+ virtual UpWall (*PlayerData) = 0
+ virtual DownWall (*PlayerData) = 0
+ virtual RightWall (*PlayerData) = 0
+ virtual LeftWall (*PlayerData) = 0
+ LeftWall(*Coordinate)

FixableRoom

· Allows for a declaration of a room which is broken on game start and requires items to fix. This inherits and defines it's variables and functions form the environment class

Variables

#fixed: bool = false - initially set to false, allows for the room to be checked if fixed, thai will allow for certain interaction inside the game, such as if the engine room is fixed then the sub can move, and new ocean terrain can be generated

#reqItems: Items - the required items to fix

description : string

inventory : Inventory

Member functions:

+ Enviorment( )
+ ~Enviorment()
+ SetDescription(string)
+ GetInfo() : string
+ GetInventory(): * Inventory
+ FixRoom(*Inventory) : bool
+ RoomFixed() : bool
+ NorthWall(*Coordinate)
+ SouthWall(*Coordinate)
+ RightWall(*Coordinate)
+ LeftWall(*Coordinate)

Ocean

· A specialized version of environment witch allows for the spawning of sea creatures like sharks

Variables

- SpawnChance : const double the spawn chance which a sea creature will spawn

Member functions:

+Ocean(double)
+ ~Ocean()
+ GetSpawnChance() : double

·　　Allow for the creation of an airlock room, witch will ensure the player has the proper gear to enter the ocean.

Variables

 # contents : inventory

Member functions:

+ ZoneEntrence(string[] Interactson ,*inventory)

+ ~ZoneEntrence()

+ EnterSea() – checks to make sure the player wants to enter the see from the ship well wearing the proper gear/equipment

+ EnterShip()

GameMap

·　　Holds information that relates to map and interaction with it. this also checks the players cords to see what room they are in/what they can do There are two maps ocean and submarine.

Variables

- gameMap : vector<Enviroment*>

Member functions:

+ GameMap()

+ ~GameMap()

+ CreateMap()

+ DestroyMap()

+ GetRoom(*Coordinate): *Enviorment

+ GetNPC(*Coordinate) : *Environment

 Coordinate

·　　Cordnates of the player, allows them to be tracked throughout the game

Variables

- xCord : int

- yCord : int

Member functions:

+ coordinate(int x, int y)

+ SetX(int)

+ SetY(int)

+ GetY() : int

+ GetX() : int

Constants:

+ enum Rooms {

 HallWay,

 Reactor,

```
  Other
};
```

PlayerData

· The games player data, allows tracking of information involving the player

Variables

- cords : Coordinate
- Inventory : Inventory
- score : int
- fixedReactor : bool = false

Member functions:

+ PlayerData()
+ ~PlayerData()
+ GetInventory() : *Inventory
+ GetCords() : *Coordinate
+ SetFixed(bool);

//+ GetScore(): int
//+ UpdateO2(int)
//+ GameOver(Int death):ostream

Inventory

· Manages inventory management for locations, characters and player

Variables

- contents :map<string,Item>

Member functions:

+ Inventory(map<string,Item>)
+ ~Inventory()
+ Compare(Item) : bool
+ AddItem(Item)
+ GetItem(string): Item
+ RemoveItem(string)
+ ListContents()
+ Transfer(*Inventory, string)
+ ListContents()

Item

· A base item definition for the game, there is a child call tool which expanded upon it

Variables
- itemname : string
- description : string
Member functions:
+Item(string itemName, string desc)

+Item(string itemName)

+ Item()

+ ~Item()

+ SetScore(int)

+ GetName() : string

+ GetDescription() : String

+ GetScore() : Int

Interaction

- Handles the definitions for the games various interactions, this is the base class which has through children who inherit from it.

+ Interaction()

+ ~Interaction()

+ ProcessMapInput(const &string,
*GameMap, *PlayerData) : bool

+ ProcessInventoryInput(const &string,
*GameMap, *PlayerData) : bool

+ North(*GameMap, *PlayerData)

+ East(*GameMap, *PlayerData)

+ South(*GameMap, *PlayerData)

+ West(*GameMap, *PlayerData)

+ Talk(*Npc) : String

+ PickUp(*PlayerData, *GameMap, const string)

+ Drop(*PlayerData, *GameMap, const string)

+ Look(*GameMap, *PlayerData)

+ Fix(*GameMap,*PlayerData)

+ ListInventory(Inventory)

+ ListInventory(*Inventory)

UnderUs

- The game's ui/interface. Takes in input and outputs various things to the user. It's really the manger for the game

Variables

- running: bool = false
- gameMap : GameMap
- player : PlayerData
- interaction : Interaction
- input : string

Member functions:

+ MainMenu()
+ DisplayBackStory()
+ Run()
+ ListCommands()
+ Help()
+ Exit()
+ GetInput() : string
+ ProcessInput(string)
+ UpperToLower(char) : char

Exceptions

· The game's error handling class.

Member Functions:

+Invalid_parameter_error(const char* errMessege)
+Calculation_error(const char* errMessage)

Constants

· Class for room enumeration.

Member Functions:

+Invalid_parameter_error(const char* errMessege)
+Calculation_error(const char* errMessage)

Game Map

Infirmary (0)	Top AirLock (1)	Barracks (2)
Hallway End (3)	Elevator (4)	Bridge (5)
Reactor (6)	AirLock (7)	Storage (8)
Reef (9)	SeaLock (10)	Cave (11)