TEAM: AFRICAN GREY PARROTS

Pandemic in C++

Prepared for: Dr. Nora Houari

Prepared by: Tse-Chun Lau (29676279),
The Phi Nguyen (40015643),
Matthew Chrobak (27381794),
Nahian Pathan (27105827)

Project Version: Final Build

April 18, 2017

EXECUTIVE SUMMARY

Objective

This project is based on re-creating the popular board game “PANDEMIC” coded from ground up using the high-level object-oriented programming language, C++. Throughout the lifetime of this project, C++ is used and enhanced for a robust and equally enjoyable game. It is now playable from any computer (Windows and Mac). It is expected to be achieved with a mindset of high cohesion and low coupling.

This is the final build ready for public consumption.

Goals

This final build, can be played from start to finish with all the all actions (please refer to user manual). This version should enable two to four players to play, including-

- Performing 4 different actions
- Once 4 actions are completed, player automatically picks up two player cards from the deck of the board.
- Cities are also automatically infected (cubes placed) depending on the infection cards picked up from the deck.

The actions are recorded and returned with appropriate information using GUI.

Solution

The final build of Pandemic is achieved through multiple classes in a hierarchical style, involving many has-a and is-a relations among those. A variation of MVC (no observer pattern) is implemented for receiving inputs from the user and representing the game. Other design pattern used in the program includes Singleton,_____, Strategy and Decorator.

Project Outline

- Description of the setup
- Game dynamic
- Source files and its corresponding header files’ name and description
- Library(s) used
- Conclusion
- UML diagram

GAME SETUP, CLASSES USED AND MORE

- **Start game**

The game is loaded from a .exe file and presented with GUI, implemented using SFML graphics library. This calls the main() method which runs the game. The game is divided into 3 states which drive the game accordingly -

1. MainMenu - Presents the player with main menu for loading or creating a game
2. InGame - Once in game, players perform series of actions
3. Closed - Ending the game and saving the stats

- **Game dynamic**

For the final build the game starts with the default map which resembles the real-life board, containing cities and other indicators such an epidemic marker.

Once the players are in-game, the instances of the player class representing the players are initialized with random role cards. Players also get 7 PlayerCards to use during actions in-compliance with the rules rules. The player can make 4 moves, selected from the sidebar, and eventually end his/her turn by picking up up two Player Cards and infecting two cities with Infection Cards. Then the second player starts a new turn. During the turns, players can see the pawns move (indicating the current location on the map), and see changes on the infection marker and epidemic rate of the game as the game progresses.

- **Exception Handling**

We used try/catch statements to handle invalid input in the Event Card actions, and in the situation of invalid file formats in the Board Builder.

- **Source files and its corresponding files**

To make the program easier to manage and build, starting with build 1, software architectural model MVC is implemented. Below are the some of the rather more important classes used to build the game and are discussed with compliance to MVC model rules -

PART OF MVC	NOTABLE SOURCE FILES	CORRESPONDING HEADER FILES	Note
Model	Player.cpp	Player.h	Creates players.
	City.cpp	City.h	Creates cities, containing attributes such as current number of disease cubes and a boolean for indicating existence of research centre.
	RoleCard.cpp	RoleCard.h	Creates RoleCard for players which decides various differences in performing actions for said player.
	PlayerCard.cpp	PlayerCard.h	Class at the top of the hierarchy which has a 'is-a' relationship with the type of PlayerCards such as CityCard(.cpp/.h), EventCard(.cpp/.h) and so on.
	ArrayGraph.cpp	ArrayGraph.h	Adds nodes to Graph, used to represent the map.
View	SFML files including- SfmlSystem.cpp SfmlSurfaceManager.cpp	SfmlSystem.h SfmlSurfaceManager.h	Classes used to derive the VIEW part of the game using the popular graphics library: SFML*. (*More on it later)
	GameRenderer.cpp	GameRenderer.h	Renders the key elements of the game such as the background, cities and infection marker
	CommonContext.cpp	CommonContext.h	Manages the references of the graphical elements on the current board, e.g shapes and colours.

PART OF MVC	NOTABLE SOURCE FILES	CORRESPONDING HEADER FILES	Note
	GraphicsManager.cpp	GraphicsManger.h	To tie up the above classes and initialized and generate appropriate graphical elements. All the methods in the files are carefully selected to be static.
CONTROLLER	UIElement.cpp (among other UI cpp files)	UIElement.h	Registers clicks from the user.
	Buttons.cpp	Buttons.h	Responsible for executing codes when a player action is to be used.

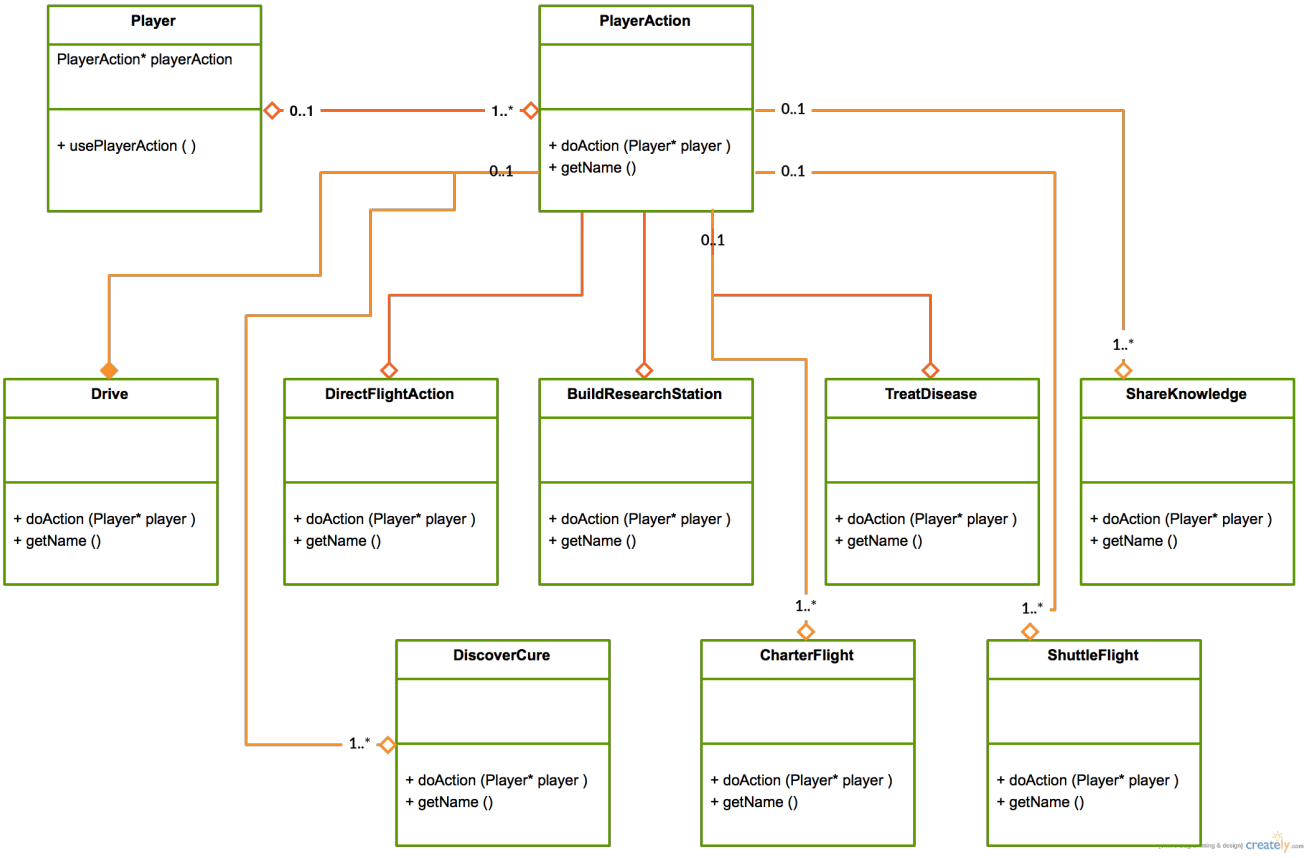
OTHER PATTERNS USED (UML DIAGRAM)

strategy pattern for player actions -

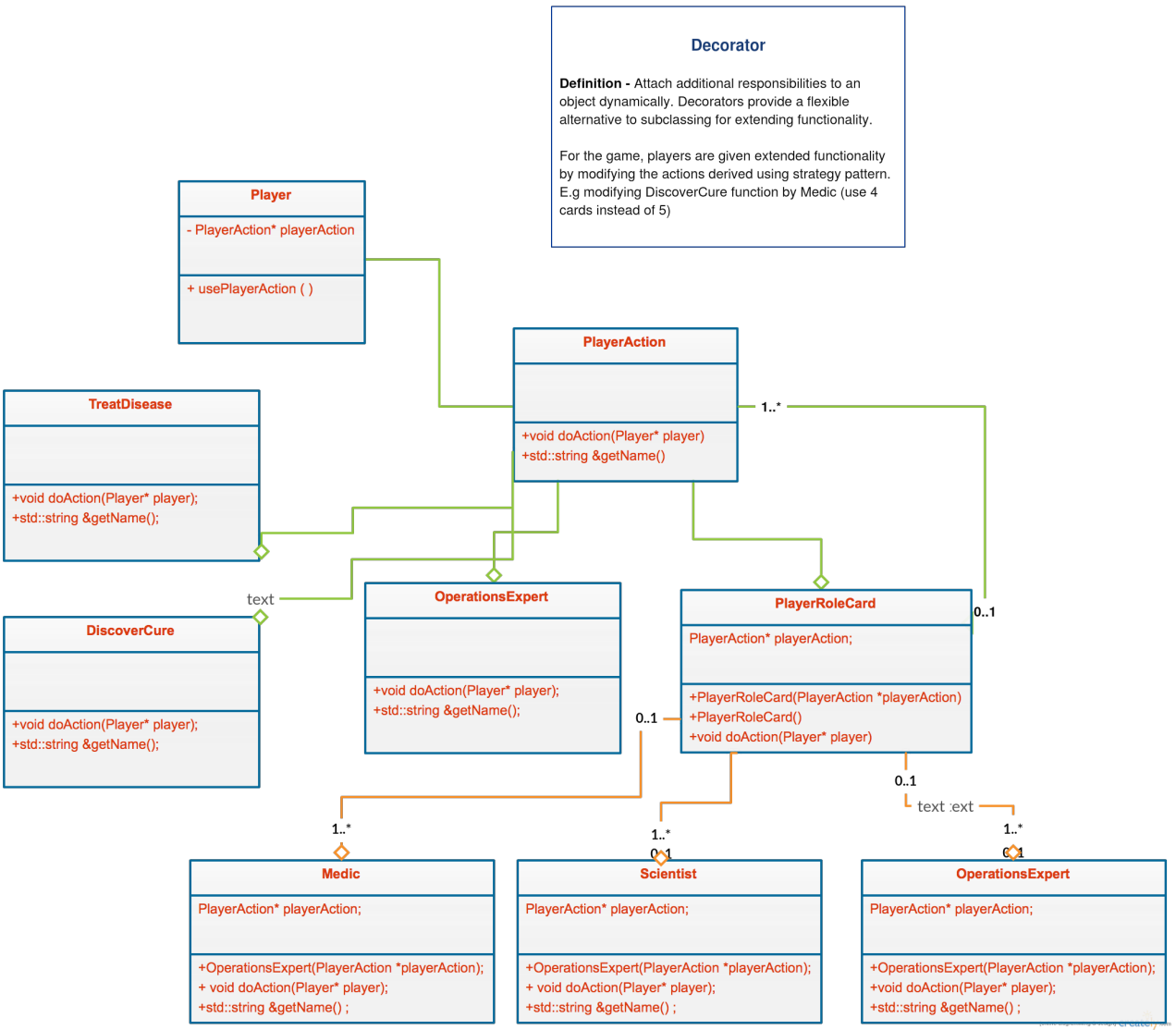
Strategy

Definition - Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

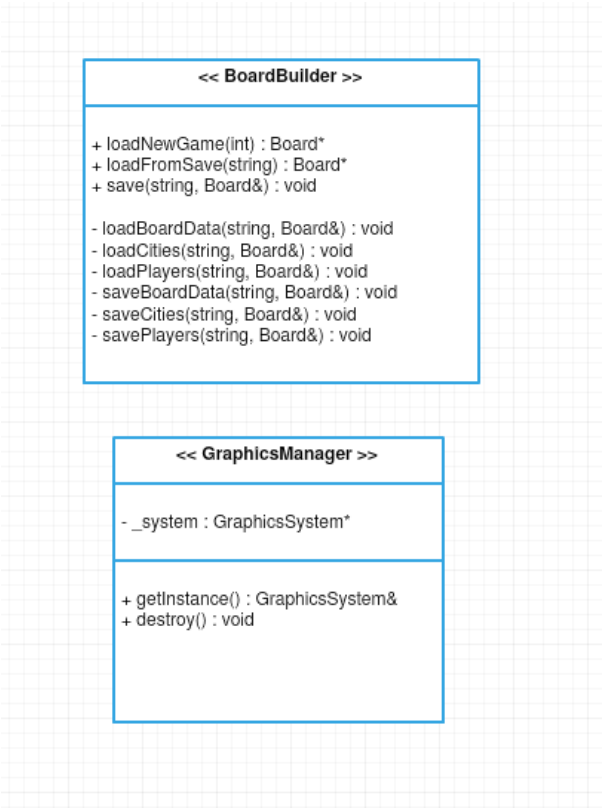
The game uses the player's different of types of strategy a.k.a the player actions that could be performed and switched dynamically.



Decorator pattern for modifying player actions -



BUIDER PATTERN - FOR LOADING AND SAVING
SINGLETON PATTER - FOR HAVING ONE INSTANCE OF THE GAME



REQUIRED LIBRARIES FOR IMPLEMENTATION

- **Libraries**

GUI - SFML

The main library used for the project is “Simple and Fast Multimedia Library” also known as SFML. It an API which provides easy access to multimedia contents in computers. It is extensively used through out the project for easy and enjoyable playability for the user. This also gives a great view of the current status of the game, which in-turn let users make better decision with playing the game.

UTILITY - BOOST

The most popular peer-to-peer reviewed library for C++ - Boost, is used extensively in the game for file system management. Played a major role is making the game cross platform.

- **Container**

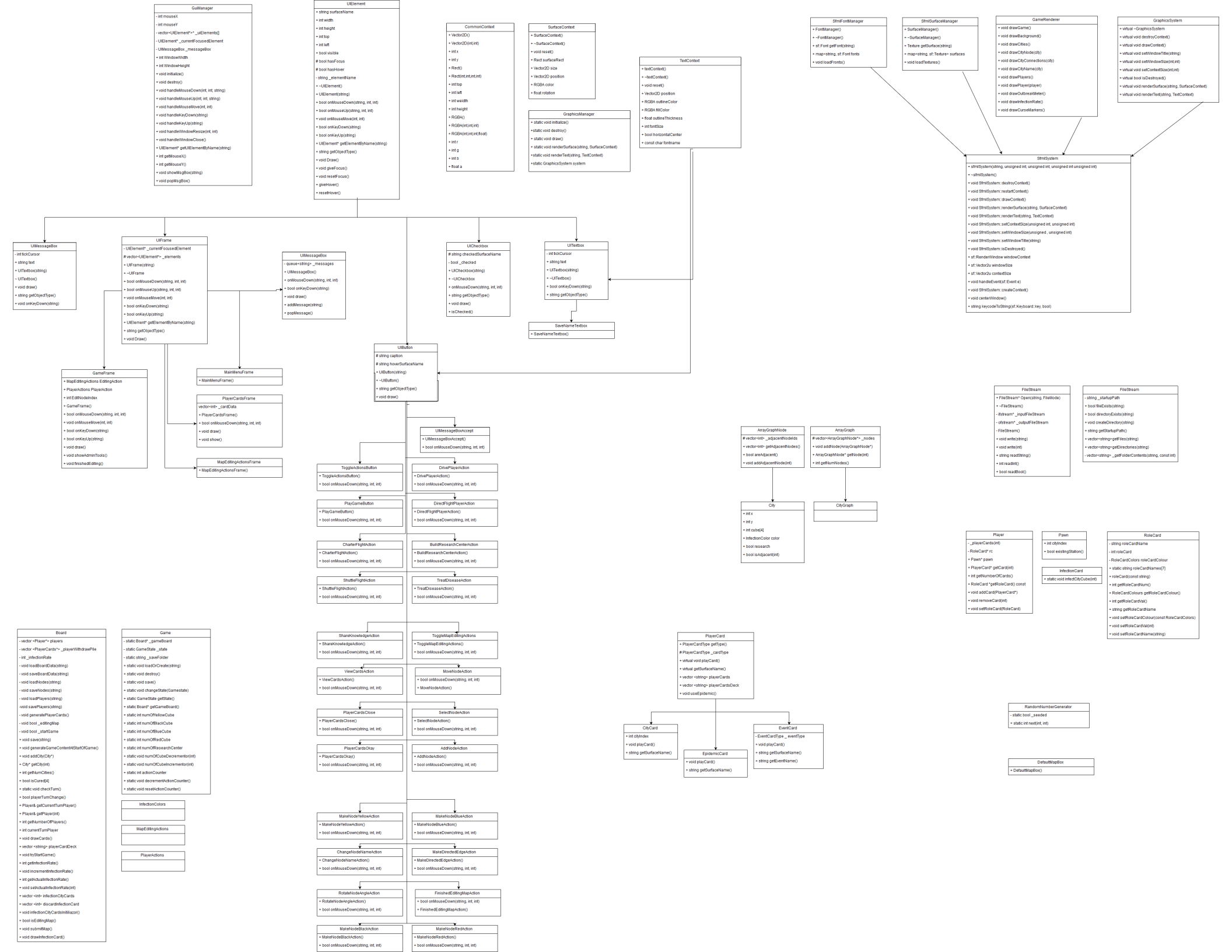
Vectors were used through out the game, for it’s easy implementation and ease of use. Since vector can change size dynamically and has using built-in methods such size(), among others , it was an easy choice to make.

CONCLUSION

The game is now fully functional and most of the actions are ready to be used (ShareKnowledge is still in the works)

The main issue with the game was the lack of knowledge about the MVC architecture and ultimately resulting in a not using Observer pattern. Because of that the implementation of logic was slightly more harder and took longer time. But the implementation of the other patterns helped make the game robust.

Below is the UML diagram of the rest of the system (please zoom-in for a better view)->


GuiManager
- int mouseX
- int mouseY
- vector<UIElement*>* _uiElements[]
- UIElement* _currentFocusedElement
- UIMessageBox _messageBox
+ int WindowWidth
+ int WindowHeight
+ void initialize()
+ void destroy()
+ void handleMouseDown(int, int, string)
+ void handleMouseUp(int, int, string)
+ void handleMouseMove(int, int)
+ void handleKeyDown(string)
+ void handleKeyUp(string)
+ void handleWindowResize(int, int)
+ void handleWindowClose()
+ UIElement* getUIElementByName(string)
+ int getMouseX()
+ int getMouseY()
+ void showMsgBox(string)
+ void popMsgBox()
UIElement
+ string surfaceName
+ int width
+ int height
+ int top
+ int left
+ bool visible
# bool hasFocus
# bool hasHover
- string _elementName
+ ~UIElement()
+ UIElement(string)
+ bool onMouseDown(string, int, int)
+ bool onMouseUp(string, int, int)
+ void onMouseMove(int, int)
+ bool onKeyDown(string)
+ bool onKeyUp(string)
+ UIElement* getElementByName(string)
+ string getObjectType()
+ void Draw()
+ void giveFocus()
+ void resetFocus()
+ giveHover()
+ resetHover()
CommonContext
+ Vector2D()
+ Vector2D(int,int)
+ int x
+ int y
+ Rect()
+ Rect(int,int,int,int)
+ int top
+ int left
+ int weidth
+ int height
+ RGBA()
+ RGBA(int,int,int)
+ RGBA(int,int,int,float)
+ int r
+ int g
+ int b
+ float a
SurfaceContext
+ SurfaceContext()
+ ~SurfaceContext()
+ void reset()
+ Rect surfaceRect
+ Vector2D size
+ Vector2D position
+ RGBA color
+ float rotation
GraphicsManager
+ static void initialize()
+static void destroy()
+ static void draw()
+ static void renderSurface(string, SurfaceContext)
+static void renderText(string, TextContext)
+static GraphicsSystem system
TextContext
+ textContext()
+ ~textContext()
+ void reset()
+ Vector2D position
+ RGBA outlineColor
+ RGBA fillColor
+ float outlineThickness
+ int fontSize
+ bool horizontalCenter
+ const char fontname
SfmlFontManager
+ FontManager()
+ ~FontManager()
+ sf::Font getFont(string)
+ map<string, sf::Font fonts
+ void loadFronts()
SfmlSurfaceManager
+ SurfaceManager()
+ ~SurfaceManager()
+ Texture getSurface(string)
+ map<string, sf::Texture> surfaces
+ void loadTextures()
GameRenderer
+ void drawGame()
+ void drawBackground()
+ void drawCities()
+ void drawCityNode(city)
+ void drawCityConnections(city)
+ void drawCityName(city)
+ void drawPlayers()
+ void drawPlayer(player)
+ void drawOutbreakMeter()
+ void drawInfectionRate()
+ void drawCurseMarkers()
GraphicsSystem
+ virtual ~GraphicsSystem
+ virtual void destroyContext()
+ virtual void drawContext()
+ virtual void setWindowTitle(string)
+ virtual void setWindowSize(int,int)
+ virtual void setContextSize(int,int)
+ virtual bool isDestroyed()
+ virtual void renderSurface(string, SurfaceContext)
+ virtual void renderText(string, TextContext)
SfmlSystem
+ sfmlSystem(string, unsigned int, unsigned int, unsigned int unsigned int)
+ ~sfmlSystem()
+ void SfmlSystem::destroyContext()
+ void SfmlSystem::restartContext()
+ void SfmlSystem::drawContext()
+ void SfmlSystem::renderSurface(string, SurfaceContext)
+ void SfmlSystem::renderText(string, TextContext)
+ void SfmlSystem::setContextSize(unsigned int, unsigned int)
+ void SfmlSystem::setWindowSize(unsigned , unsigned int)
+ void SfmlSystem::setWindowTitle(string)
+ void SfmlSystem::isDestroyed()
+ sf::RenderWindow windowContext
+ sf::Vector2u windowSize
+ sf::Vector2u contextSize
+ void handleEvent(sf::Event e)
+ void SfmlSystem::createContext()
+ void centerWindow()
+ string keycodeToString(sf::Keyboard::key, bool)
UIMessageBox
- int tickCursor
+ string text
+ UITextbox(string)
~ UITextbox()
+ void draw()
+ string getObjectType()
+ void onKeyDown(string)
UIFrame
- UIElement* _currentFocusedElement
vector<UIElement*> _elements
+ UIFrame(string)
+ ~UIFrame
+ bool onMouseDown(string, int, int)
+ bool onMouseUp(string, int, int)
+ void onMouseMove(int, int)
+ bool onKeyDown(string)
+ bool onKeyUp(string)
+ UIElement* getElementByName(string)
+ string getObjectType()
+ void Draw()
UIMessageBox
- queue<string> _messages
+ UIMessageBox()
+ onMouseDown(string, int, int)
+ bool onKeyDown(string)
+ void draw()
+ addMessage(string)
+ popMessage()
UICheckbox
string checkedSurfaceName
- bool _checked
+ UICheckbox(string)
+ ~UICheckbox
+ onMouseDown(string, int, int)
+ string getObjectType()
+ void draw()
+ isChecked()
UITextbox
- int tickCursor
+ string text
+ UITextbox(string)
+ ~UITextbox()
+ bool onKeyDown(string)
+ string getObjectType()
SaveNameTextbox
+ SaveNameTextbox()
GameFrame
+ MapEditingActions EditingAction
+ PlayerActions PlayerAction
+ int EditNodeIndex
+ GameFrame()
+ bool onMouseDown(string, int, int)
+ void onMouseMove(int, int)
+ bool onKeyDown(string)
+ bool onKeyUp(string)
+ void draw()
+ void showAdminTools()
+ void finishedEditing()
MainMenuFrame
+ MainMenuFrame()
PlayerCardsFrame
vector<int> _cardData
+ PlayerCardsFrame()
+ bool onMouseDown(string, int, int)
+ void draw()
+ void show()
MapEditingActionsFrame
+ MapEditingActionsFrame()
UIButton
string caption
string hoverSurfaceName
+ UIButton(string)
+ ~UIButton()
+ string getObjectType()
+ void draw()
UIMessageBoxAccept
+ UIMessageBoxAccept()
+ bool onMouseDown(string, int, int)
ToggleActionsButton
+ ToggleActionsButton()
+ bool onMouseDown(string, int, int)
DrivePlayerAction
+ DrivePlayerAction()
+ bool onMouseDown(string, int, int)
PlayGameButton
+ PlayGameButton()
+ bool onMouseDown(string, int, int)
DirectFlightPlayerAction
+ DirectFlightPlayerAction()
+ bool onMouseDown(string, int, int)
CharterFlightAction
+ CharterFlightAction()
+ bool onMouseDown(string, int, int)
BuildResearchCenterAction
+ BuildResearchCenterAction()
+ bool onMouseDown(string, int, int)
ShuttleFlightAction
+ ShuttleFlightAction()
+ bool onMouseDown(string, int, int)
TreatDiseaseAction
+ TreatDiseaseAction()
+ bool onMouseDown(string, int, int)
ShareKnowledgeAction
+ ShareKnowledgeAction()
+ bool onMouseDown(string, int, int)
ToggleMapEditingActions
+ ToggleMapEditingActions()
+ bool onMouseDown(string, int, int)
ViewCardsAction
+ ViewCardsAction()
+ bool onMouseDown(string, int, int)
MoveNodeAction
+ bool onMouseDown(string, int, int)
+ MoveNodeAction()
PlayerCardsClose
+ PlayerCardsClose()
+ bool onMouseDown(string, int, int)
SelectNodeAction
+ SelectNodeAction()
+ bool onMouseDown(string, int, int)
PlayerCardsOkay
+ PlayerCardsOkay()
+ bool onMouseDown(string, int, int)
AddNodeAction
+ AddNodeAction()
+ bool onMouseDown(string, int, int)
MakeNodeYellowAction
+ MakeNodeYellowAction()
+ bool onMouseDown(string, int, int)
MakeNodeBlueAction
+ MakeNodeBlueAction()
+ bool onMouseDown(string, int, int)
ChangeNodeNameAction
+ ChangeNodeNameAction()
+ bool onMouseDown(string, int, int)
MakeDirectedEdgeAction
+ MakeDirectedEdgeAction()
+ bool onMouseDown(string, int, int)
RotateNodeAngleAction
+ RotateNodeAngleAction()
+ bool onMouseDown(string, int, int)
FinishedEditingMapAction
+ bool onMouseDown(string, int, int)
+ FinishedEditingMapAction()
MakeNodeBlackAction
+ MakeNodeBlackAction()
+ bool onMouseDown(string, int, int)
MakeNodeRedAction
+ MakeNodeRedAction()
+ bool onMouseDown(string, int, int)
Board
- vector <Player*> players
- vector <PlayerCards*> _playerWithdrawPile
- int _infectionRate
- void loadBoardData(string)
- void saveBoardData(string)
- void loadNodes(string)
- void saveNodes(string)
- void loadPlayers(string)
-void savePlayers(string)
- void generatePlayerCards()
- void bool _editingMap
- void bool _startGame
+ void save(string)
+ void generateGameContentAtStartOfGame()
+ void addCity(City*)
+ City* getCity(int)
+ int getNumCities()
+ bool isCured[4]
+ static void checkTurn()
+ bool playerTurnChange()
+ Player& getCurrentTurnPlayer()
+ Player& getPlayer(int)
+ int getNumberOfPlayers()
+ int currentTurnPlayer
+ void drawCards()
+ vector <string> playerCardDeck
+ void tryStartGame()
+ int getInfectionRate()
+ void incrementInfectionRate()
+ int getActualInfectionRate()
+ void setActualInfectionRate(int)
+ vector <int> infectionCityCards
+ vector <int> discardInfectionCard
+ void infectionCityCardsInitIlizator()
+ bool isEditingMap()
+ void submitMap()
+ void drawInfectionCard()
Game
- static Board* _gameBoard
- static GameState _state
- static string _saveFolder
+ static void loadOrCreate(string)
+ static void destroy()
+ static void save()
+ static void changeState(Gamestate)
+ static GameState getState()
+ static Board* getGameBoard()
+ static int numOfYellowCube
+ static int numOfBlackCube
+ static int numOfBlueCube
+ static int numOfRedCube
+ static int numOfResearchCenter
+ static void numOfCubeDecrementor(int)
+ static void numOfCubeIncrementor(int)
+ static int actionCounter
+ static void decrementActionCounter()
+ static void resetActionCounter()
InfectionColors
MapEditingActions
PlayerActions
FileStream
+ FileStream* Open(string, FileMode)
+ ~FileStream()
- ifstream* _inputFileStream
- ofstream* _outputFileStream
- FileStream()
+ void write(string)
+ void write(int)
+ string readString()
+ int readInt()
+ bool readBool()
FileStream
- string _startupPath
+ bool fileExists(string)
+ bool directoryExists(string)
+ void createDirectory(string)
+ string getStartupPath()
+ vector<string>getFiles(string)
+ vector<string>getDirectories(string)
- vector<string> _getFolderContents(string, const int)
ArrayGraphNode
vector<int> _adjacentNodeIds
+ vector<int> getAdjacentNodes()
+ bool areAdjacent()
+ void addAdjacentNode(int)
ArrayGraph
vector<ArrayGraphNode*> _nodes
+ void addNode(ArrayGraphNode*)
+ ArrayGraphNode* getNode(int)
+ int getNumNodes()
City
+ int x
+ int y
+ int cube[4]
+ InfectionColor color
+ bool research
+ bool isAdjacent(int)
CityGraph
Player
- _playerCards(int)
- RoleCard* rc
+ Pawn* pawn
+ PlayerCard* getCard(int)
+ int getNumberOfCards()
+ RoleCard *getRoleCard() const
+ void addCard(PlayerCard*)
+ void removeCard(int)
+ void setRoleCard(RoleCard)
Pawn
+ int cityIndex
+ bool existingStation()
InfectionCard
+ static void infectCityCube(int)
RoleCard
- string roleCardName
- int roleCard
- RoleCardColors roleCardColour
+ static string roleCardNames[7]
+ roleCard(const string)
+ int getRoleCardNum()
+ RoleCardColours getRoleCardColour()
+ int getRoleCardVal()
+ string getRoleCardName
+ void setRoleCardColour(const RoleCardColors)
+ void setRoleCardVal(int)
+ void setRoleCardName(string)
PlayerCard
+ PlayerCardType getType()
# PlayerCardType _cardType
+ virtual void playCard()
+ virtual getSurfaceName()
+ vector <string> playerCards
+ vector <string> playerCardsDeck
+ void useEpidemic()
CityCard
+ int cityIndex
+ void playCard()
+ string getSurfaceName()
EpidemicCard
+ void playCard()
+ string getSurfaceName()
EventCard
- EventCardType _ eventType
+ void playCard()
+ string getSurfaceName()
+ string getEventName()
RandomNumberGenerator
- static bool _seeded
+ static int next(int, int)
DefaultMapBox
+ DefaultMapBox()