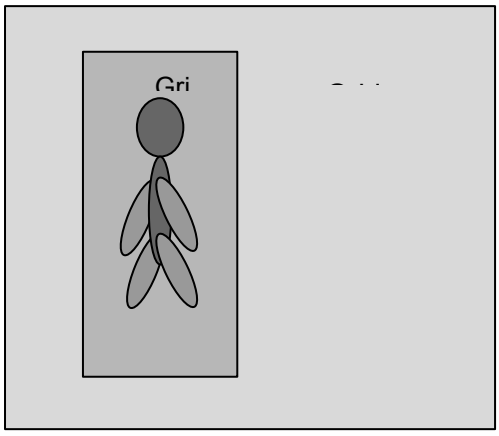
Implementation Process

After determining the functional and non-functional requirements ,we decided to use mvc and singleton design patterns. However, after we started to forming solution diagram we noticed layer design pattern would be better for our plans. In design process, we did not take into consideration the specifications of the language(C#) and the IDE we used(Visual Studio) our first layer, the graphical user interface changed more than we anticipated. Nevertheless, layer design pattern provide us a relatively easier and simpler change process. On the other hand we had to do some experiments to determine what game settings would be the best. Again the layer pattern allow us to do rapid and systematic changes without worrying about other layers. Moreover one of the reason of the design changes was the over-engineering, we created unnecessary classes due to our lack of experience. Moreover, as another benefit of layer pattern, we divided the implementation based on layers first and it provide us a relatively easy merge process.

Any of the group members developed a project with C#, because of that we were unable to foresee possible difficulties due to the conflicts between our design and language. For example event handling system in C# forced us to change our design through implementation.

One of the biggest challenges we experienced was the choosing and implementing graphical user interface. At first stage, Run To Live was designed to use WPF and we considered that WPF can handle player animation more powerful and with smoothly movements. To achive this goal, User Control of WPF is started to written as WPF User Control. In this control, player is implemented with walking man and this walk is done by Animation types of System.Media library. The way of implementing User Control is like that;

On the Grid component, player is created by composed of ellipses and specified angles provide player's legs and arms. Also this player is inside another grid.



This man has walking animation and it is done by changing the legs and arms angles in a harmonious way. According to mouse click, event handler gets the position and man started to walk that point on the bottom grid.

Problem
In the game screen, countries are implemented on the buttons. So that, via clicking every button, you can see the information of the given country. Problem arises here; since player should move over these buttons, grid that include player panel should be on top of the buttons. In this situation, we cannot handle with the problem of overlapping buttons and player grid. Why this situation crates a problem? Because, when user control is placed on the buttons of country, click events of buttons didn't work. Also, if there were no grid that player is placed inside, coordination of player headed cannot be determined. This overlapping issue creates a problem and in a limited time, we cannot solve overlapping components problem.

Reason
Wrong design decision is the main reason for us to facing this problem. Also another reason is we didn't have project manager to keep track of every work done and sometimes members of group wrote their code independently that means we couldn't use Github actively. So after every change in code, catching last version of project is sometimes forced us.

What have we learned?

We learned that project management is important then we thought. Even the our project was not large working as a group efficiently requires a better workflow understanding. For example, we did not define the connection points of the layers and that caused additional workload for us.

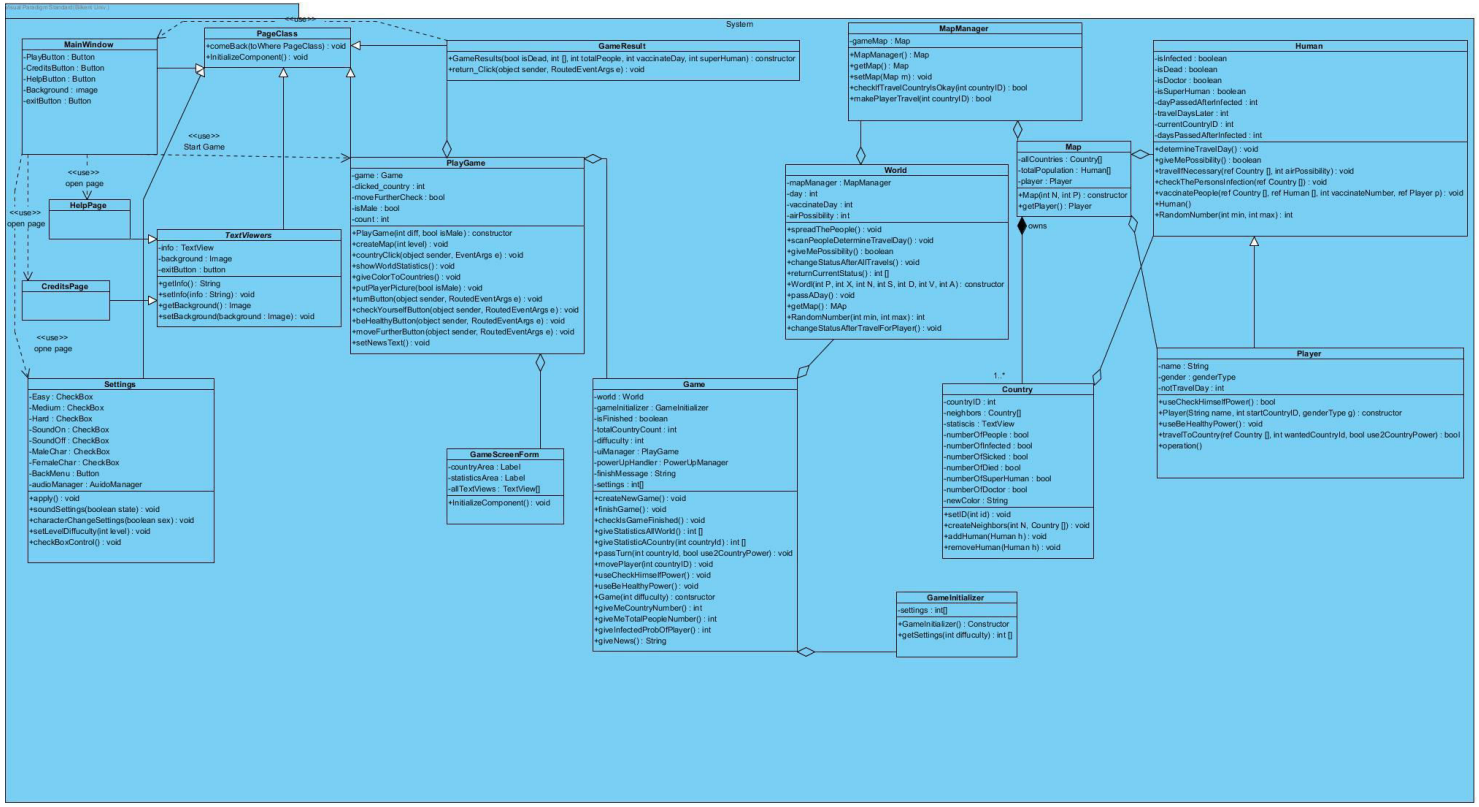
We learned we should have used the project management tools like JIRA or version control tools like GitHub. Because we experienced some difficulties at working on the same version. Sometimes 2 people changed similar things without noticing or someone worked on the old version. Moreover, JIRA could allow us to see what other members are doing and follow deadlines better.

We learned the importance of the documentation. We mostly wrote our documentation afterwards. We could be more efficient if every member wrote the documentation of his/her part then merge the documentation.

We learned the importance of the designing and implementing proper tests. Our third layer, the game entities contains most of the calculations about the game. Whenever we faced a difficulty sourced from that layer we needed to check a very complex and hard to understand code to find what causes the bug. However, with proper tests we could narrow the code we should check and find the problem much easier.

Parts that not implemented yet

Game navigation: Visiting settings,help or credits page from the running game screen is not implemented yet.


System
MainWindow
-PlayButton : Button
-CreditsButton : Button
-HelpButton : Button
-Background : image
-exitButton : Button
PageClass
+comeBack(toWhere PageClass) : void
+InitializeComponent() : void
GameResult
+GameResults(bool isDead, int [], int totalPeople, int vaccinateDay, int superHuman) : constructor
+return_Click(object sender, RoutedEventArgs e) : void
MapManager
-gameMap : Map
+MapManager() : Map
+getMap() : Map
+setMap(Map m) : void
+checkIfTravelCountryIsOkay(int countryID) : bool
+makePlayerTravel(int countryID) : bool
Human
-isInfected : boolean
-isDead : boolean
-isDoctor : boolean
-isSuperHuman : boolean
-dayPassedAfterInfected : int
-travelDaysLater : int
-currentCountryID : int
-daysPassedAfterInfected : int
+determineTravelDay() : void
+giveMePossibility() : boolean
+travelIfNecessary(ref Country [], int airPossibility) : void
+checkThePersonsInfection(ref Country []) : void
+vaccinatePeople(ref Country [], ref Human [], int vaccinateNumber, ref Player p) : void
+Human()
+RandomNumber(int min, int max) : int
<<use>>
Start Game
<<use>>
open page
<<use>>
open page
<<use>>
opne page
HelpPage
CreditsPage
TextViewers
-info : TextView
-background : Image
-exitButton : button
+getInfo() : String
+setInfo(info : String) : void
+getBackground() : Image
+setBackground(background : Image) : void
PlayGame
-game : Game
-clicked_country : int
-moveFurtherCheck : bool
-isMale : bool
-count : int
+PlayGame(int diff, bool isMale) : constructor
+createMap(int level) : void
+countryClick(object sender, EventArgs e) : void
+showWorldStatistics() : void
+giveColorToCountries() : void
+putPlayerPicture(bool isMale) : void
+turnButton(object sender, RoutedEventArgs e) : void
+checkYourselfButton(object sender, RoutedEventArgs e) : void
+beHealthyButton(object sender, RoutedEventArgs e) : void
+moveFurtherButton(object sender, RoutedEventArgs e) : void
+setNewsText() : void
World
-mapManager : MapManager
-day : int
-vaccinateDay : int
-airPossibility : int
+spreadThePeople() : void
+scanPeopleDetermineTravelDay() : void
+giveMePossibility() : boolean
+changeStatusAfterAllTravels() : void
+returnCurrentStatus() : int []
+World(int P, int X, int N, int S, int D, int V, int A) : constructor
+passADay() : void
+getMap() : MAp
+RandomNumber(int min, int max) : int
+changeStatusAfterTravelForPlayer() : void
Map
-allCountries : Country[]
-totalPopulation : Human[]
-player : Player
+Map(int N, int P) : constructor
+getPlayer() : Player
owns
1..*
Player
-name : String
-gender : genderType
-notTravelDay : int
+useCheckHimselfPower() : bool
+Player(String name, int startCountryID, genderType g) : constructor
+useBeHealthyPower() : void
+travelToCountry(ref Country [], int wantedCountryId, bool use2CountryPower) : bool
+operation()
Country
-countryID : int
-neighbors : Country[]
-statiscis : TextView
-numberOfPeople : bool
-numberOfInfected : bool
-numberOfSicked : bool
-numberOfDied : bool
-numberOfSuperHuman : bool
-numberOfDoctor : bool
-newColor : String
+setID(int id) : void
+createNeighbors(int N, Country []) : void
+addHuman(Human h) : void
+removeHuman(Human h) : void
Game
-world : World
-gameInitializer : GameInitializer
-isFinished : boolean
-totalCountryCount : int
-difficulty : int
-uiManager : PlayGame
-powerUpHandler : PowerUpManager
-finishMessage : String
-settings : int[]
+createNewGame() : void
+finishGame() : void
+checkIsGameFinished() : void
+giveStatisticsAllWorld() : int []
+giveStatisticACountry(int countryId) : int []
+passTurn(int countryId, bool use2CountryPower) : void
+movePlayer(int countryID) : void
+useCheckHimselfPower() : void
+useBeHealthyPower() : void
+Game(int difficulty) : conteructor
+giveMeCountryNumber() : int
+giveMeTotalPeopleNumber() : int
+giveInfectedProbOfPlayer() : int
+giveNews() : String
GameScreenForm
-countryArea : Label
-statisticsArea : Label
-allTextViews : TextView[]
+InitializeComponent() : void
GameInitializer
-settings : int[]
+GameInitializer() : Constructor
+getSettings(int difficulty) : int []
Settings
-Easy : CheckBox
-Medium : CheckBox
-Hard : CheckBox
-SoundOn : CheckBox
-SoundOff : CheckBox
-MaleChar : CheckBox
-FemaleChar : CheckBox
-BackMenu : Button
-audioManager : AudioManager
+apply() : void
+soundSettings(boolean state) : void
+characterChangeSettings(boolean sex) : void
+setLevelDifficulty(int level) : void
+checkBoxControl() : void
WPF
PlayerWinFormUserControl
WindowsFormHost
WpfUserControl.xaml
-myGrid : Grid
-playerPath : Path
WPFUserControl
-positionOfPlayer : Point
-goToPosition : Point
-isAnimationBegin : bool
-isClick : bool
-shadowEffect : DropShadowEffect
-blurEffect : BlurBitmapEffect
+stopAnimations() : void
+initialize() : void
+myGrid_MouseLeftButtonDown(object sender, MouseEventArgs e) : void
+myGrid_MouseMove(object sender, MouseEventArgs e) : void
+path_MouseLeftButtonDown(object sender, MouseEventArgs e) : void
+myGrid_Loaded(object sender, RoutedEventArgs e) : void
+WpfUserControl_MouseLeave(object sender, MouseEventArgs e) : void
+path_MouseLeftButtonUp(object sender, MouseButtonEventArgs e) : void

How to Install

1- Clone the repository to your computer
2-Open the folder and follow the next path
3- “RunToLive\RunToLive\bin\Debug\RunToLive.exe”
4- Run the RunToLive.exe
5- Play the game.

User’s Guide

First screen

1-Credits

User can use credits button to see the info about developers.

2-Help

User can use help button to see the game rules - highly recommended to visit before starting the game

3-Settings

User can use credits button to see the info about developers.

4-Play Game

After learning the game and adjusting the settings, user can use play game button to start a new game.

Game Play Screen

1-Countries

Counry grid contains country buttons, user should press a button to choose country to travel at the end of the turn.

2-Power Ups

User can use power up buttons to use power ups(explained in help page).Power ups dont apply at the end of the turn. Thus, user should be careful before using them.

3-Information boxes

Information boxes contains useful information about whole world and a specific country.

4-End Turn Button

User should use end turn button to end the turn and travel to the selected country.