Programming 3: Technique - TIE-02402

PROJECT DOCUMENTATIONS

for

***Game ”NowItWhen*”**

Duy Vu – duy.vu@tuni.fi

Hasan Farooq – hasan.farooq@tuni.fi

1. Requirement and additional features

Minimum requirements (Max Grade 1):

- ✓ Successfully compiling game.
- ✓ Some kind of game-mechanics implemented (e.g. player turns, scoring)
- ✓ One dialog-window (e.g. Start menu to specify amount of players, goals etc.)
- ✓ At least one own actor
- ✓ UnitTests have been written at least for the Statistics class, which keeps track of the events in the game so far and if needed calculates the points.
- ✓ Final document that must contain:
    - Game-rules
    - Game-controls
    - Class responsibilities (class diagram is good tool in this)
    - Division of labor.

Intermediate requirements (Max Grade 3):

- ✓ Meets minimum requirements.
- ✓ Own graphics fully implemented by the team (no using the course library drawing)
- ✓ The playable figure can be moved on the game are. For an intermediate implementation it is enough that the figure moves immediately without any animation e.g. to the point indicated by the cursor or a fixed amount of steps to a direction shown.
- ✓ At least one additional unique actor has been implemented
- ✓ The game works
- ✓ Bonuspoints can compensate up to one grade's worth of mistakes.

Top-grade requirements (Max Grade 5):

- ✓ Must meet minimum and intermediate requirements
- ✓ At least 2 additional features.
- ✓ Bonuspoints can compensate up to one grade's worth of mistakes.

ADDITIONAL FEATURES

- ✓ Even screen updates. The game area is updated to show the changes with fixed intervals instead each triggering a separate change. The QTimerclass can be useful here.
  Description: Game timer is shown with the help of QTimer so main window is updated after fixed intervals
- ✓ Minimal screen update. When an actor's state is changed, only its surroundings are update in the game area instead of drawing the entire map.
  Description: Actors in city class are updated using their move functions only which takes the difference delta and then move the actor so rest of the scene is not touched
- ✓ Even movement of the playable figure. Instead of immediately jumping from one place to another, the figure is told to go to a place and it moves there with a suitable speed.

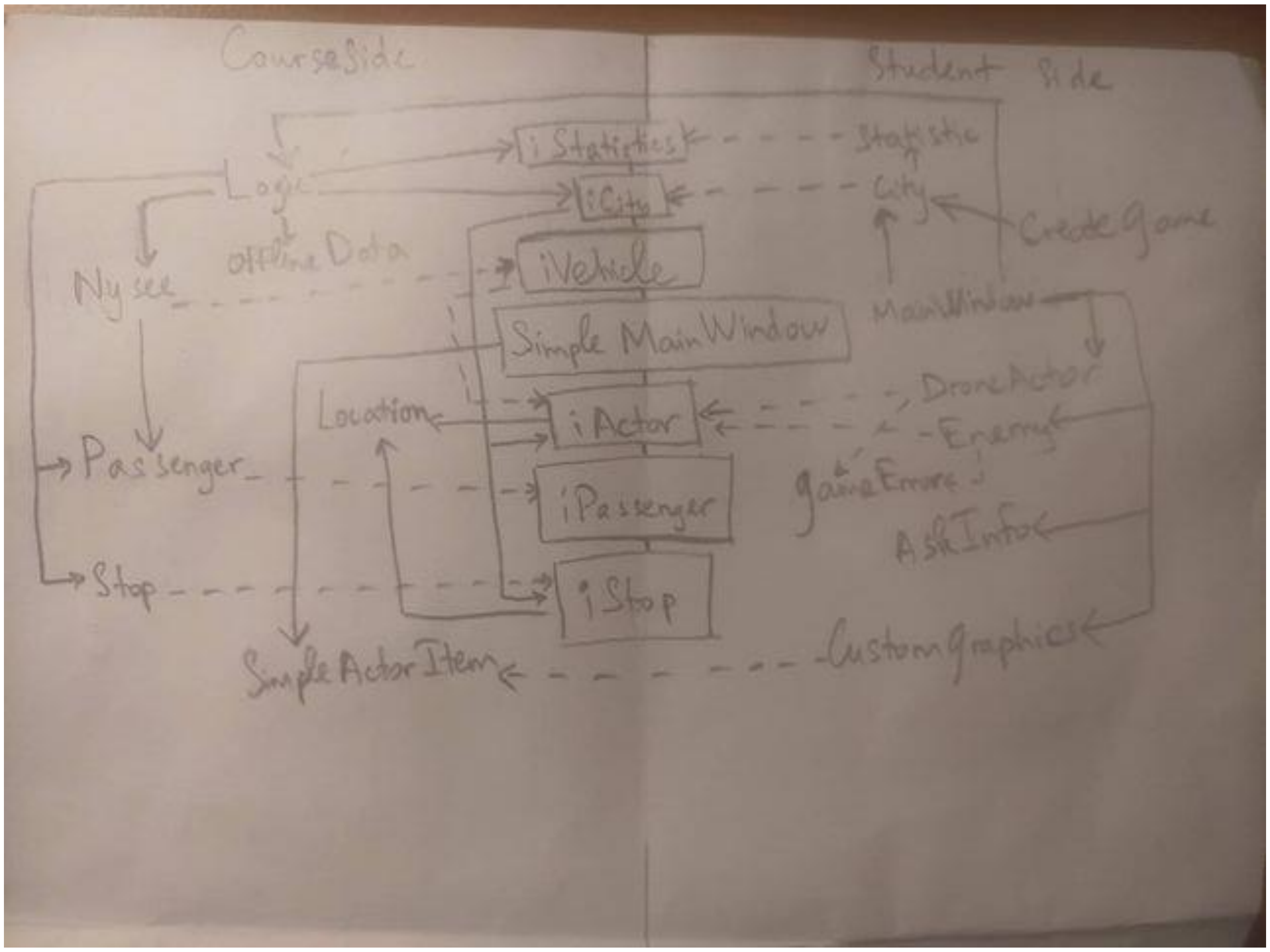
Description: The drone actor moves with the speed of mouse and does not jump unless mouse goes outside of game area while dragging the drone actor (drone cannot go outside this area)

- ✓ Following the game state. Statistics collected during the game are shown in real time in the main window.
  Description: Score is shown in real time
- ✓ Own feature. An additional feature outside this list has been implemented. Note! The feature needs to clearly require coding.
  Description: The gameplay is unique every time on game start. It was implemented in the Game/mainwindow.cpp file with the help of QRandomGenerator (line 91) and bound method (line 95) to set buses running in the morning schedule instead of night schedule in order to get more active buses.

NOT IMPLEMENTED ADDITIONAL FEATURES:

- ✓ Scrollable map. The game area is not tied to the Tampere city center. The area changes based on the movements of the playable figure. (max. 1 grade)
- ✓ Passenger amounts. The amounts of passengers on the busses and the bus stops is shown on the map for example as numbers over the icon.
- ✓ Local multiplayer. There can be several playable figures in the game (e.g. a zombie passengers and a zombie fighter) that different players can control for example one with a mouse and one with the keyboard. The points can be joint or separate. (max. 1 grade)
- ✓ Top10-list. A Top10 list is implemented into the game. The contents of the list are kept from closing the game to starting it again.
- ✓ A wide range of transport choices. The playable figure has at least three different means of transportation to move about the city.
- ✓ Updates to the playable figure. The playable figure can be updated by collecting, buying or acquiring upgrades in some other way. The upgrades effect the figures movement or attack abilities.
- ✓ Hijacking passengers. The playable figure can highjack the passengers in some way for example from disabled busses and transport them to a suitable point on the map.
- ✓ Own feature. An additional feature outside this list has been implemented. Note! The feature needs to clearly require coding.
- ✓ Ai player (max. 1 arvosana)
- ✓ Saving the game state and continuing after restart (max. 1 arvosana)

2. Struture of the software implemented



3. Responsibilities of key classes:

- Askinfo: Dialog window showing up at the beginnig of the game to ask user’s name and game level

- Mainwindow: Handling GUI of the game

- Droneactor: Playable figure of the game, used by player

- City: Handling operation of all actors during the games

- Enemy: Addtional actor chasing and attacking DroneActor

- Customgraphics: Drawing graphics for bus, passengers, or stops

- Logic: Contain basic operations handling buses, passengers and stops

- Statistic: Storing number of passengers, buses and scores

4. Project functionality

- After the player types in his/her name and chooses the game level they want to play, those 2 variables will be stored as private variables inside AskInfo class and can be accessed using function get_name() and get_level().

- MainWindow will then create map(QGraphicsScene object) and share pointers for City and Logic objects, object Statistic. It also creates and adds Drone and Enemy actors in the game area.

- Logic will then generate and move actors, buses, and bus stops. It will also make city draw those things on the game area using CustomGraphics class.

- When player control DroneActor, movement of drone is handled within DroneActor class.

- Enemy chases drone at every second via signal slot mechanism. At every second, it checks current drone's location and move towards it.

- When drone actor attacks buses, private variable score within Statistic will be updated and can be accessed by MainWindow to display points.

5. Work division (chronological order)

Hassan	Duy
Create Mainwindow	Create dialogue window
Create playable figure (DroneActor)	Implement City class
Use Logic class to create basic functional game with basic actors (buses, passengers, DroneActor)	Create additional actor (Enemy)
Create GUI for passengers, stops and buses	Make Unit Test for Statistic class
Document additional features	Finalize documentation

6. User manual – game rules

The game starts with a blank white main window saying, "Wait for user input". The user starts the game by clicking the "Start" button. Then, a dialog window pops up and the user will type his/her name and select the game level they want to play with by sliding the bar or changing the number inside the box next to the bar. There are 5 levels (1-5). This game level will determine speed of chasing Enemy (explain later). The player clicks "OK" to confirm and the game is started. If the player chooses "Exit" or click the "X" icon on the top right corner, the game will end immediately. When the game is initialized, your playable figure exists as a blue drone at the top left corner of the game area, and your enemy is a black plane sitting at the bottom right corner of the game area. Moreover, on the map, there's also bus stops , buses . Passengers can be identified as a small cyan dot next to the buses or bus stops. Besides the game area, there's a clock running next to it and displayed points gathering from. Players use left mouse button to click and drag the mouse to the bus they want to attack. When player is able to drag the drone to the bus, he/she has to hit space bar to attack. The player has to attack the buses with passengers while moving (dragging the drone) at the same time. Only after that, the bus is destroyed, and the player gain 10 points. However, the mission is not only attacking the buses, but also avoid being attacked by the Enemy. The chasing speed of the enemy depends on how the game level is chosen at the beginning of the game. The higher the game level is, the faster the Enemy. The game ends either if the player win (is able to attack 10 buses (gain 100 points)) or lose (is hit by enemy). After the game end, the map is clear, the clock stops, gained points and time are still displayed. The player has to click the top right corner button "X" to close the game and re-run the program to play the game again.

Note: The drone and enemy cannot escape the game area.

7. Known bugs or missing features

- Known bugs:

  +) When the player drags the drone to the edge of the game area, and the enemy chases and hits the edge, the enemy plane will stop chasing. But still if the drone is dragged again out of the edge, the enemy plane will keep chasing.

  +) Sometimes, the enemy go in a zig zag direction, but still towards the drone.

- Missing features: Some missing additional features are specified above.