Assignment_UserInterface Documentation

Controls:

WASD: move selection in Menu, move cursor in game
QE: switch between units, only possible when in selection state
Up, Left, Down, Right: choose different actions in game
 Up (yellow): no action
 Left (blue): change to ability state, use Ultimate ability
 Down (green): move, accept, use basic attack,
 Right (red): end turn, cancel, back
Esc: return to main menu

Keyboard control works for both players. If played with controller, there is a separation between players, so controller1 can only control player1 and vise versa

WASD -> equals Left stick
QE -> equals left and right Bumper
Up, Left, Down, Right -> equals the 4 buttons on the right side of the controller ("1, 2, 3, 4", or "X, Y, A, B", ...) where Up is the topmost button and so on
Esc -> equals "Start" button

1. Create a start screen with the FH logo (1 point)
    - FHS Logo transparent background
    - FHS Logo white background

    - *Implemented with tiled*

2. Create a main menu with at least three options (4 points).
    - Mandatory: Start, Quit
    - Third option switches to another window, e.g., credits, settings, ...
    - This window can be another state, or a state machine inside of the main menu state.

    - *MainMenu has three options*
    - *Buttons and Labels are placed in tiled... custom properties determine which Events are fired upon activation*
    - *Navigation with W and S or the Left Stick... Confirm with Space, Arrow Down or A-Button on Controller*
    - *Switches to either MainState.h/.cpp or CreditsState.h/.cpp*
    - *GUIRenderComponent.h/.cpp holds tgui::Gui Object, and IGUIWidgetComponent.h/.cpp + Derived Classes hold single Widgets*

3. Create in-game user interface showing at least two items (points collected, score of pinball player) (5 points).
    - Use an event bus to notify user interface of point changes, i.e. decouple GUI from game logic.
    - Use lambda closures or the observer pattern to implement a centralized EventBus.

  - Check the slides for implementation details of events and observer pattern.

- GUI is built in Tiled
- Eventbus implemented in Eventbus.h/.cpp
- Events can be fired from anywhere in the Code
- Events implemented in IGameEvent.h/.cpp and GameEventClasses.h/.cpp (derived Classes)
  Each Class has enum (GameEvents.h) for identification
- Listeners have to implement IEventListener.h/.cpp and override void onEvent(IGameEvent *event)
  Listeners get registered automatically when IEventListener is inherited