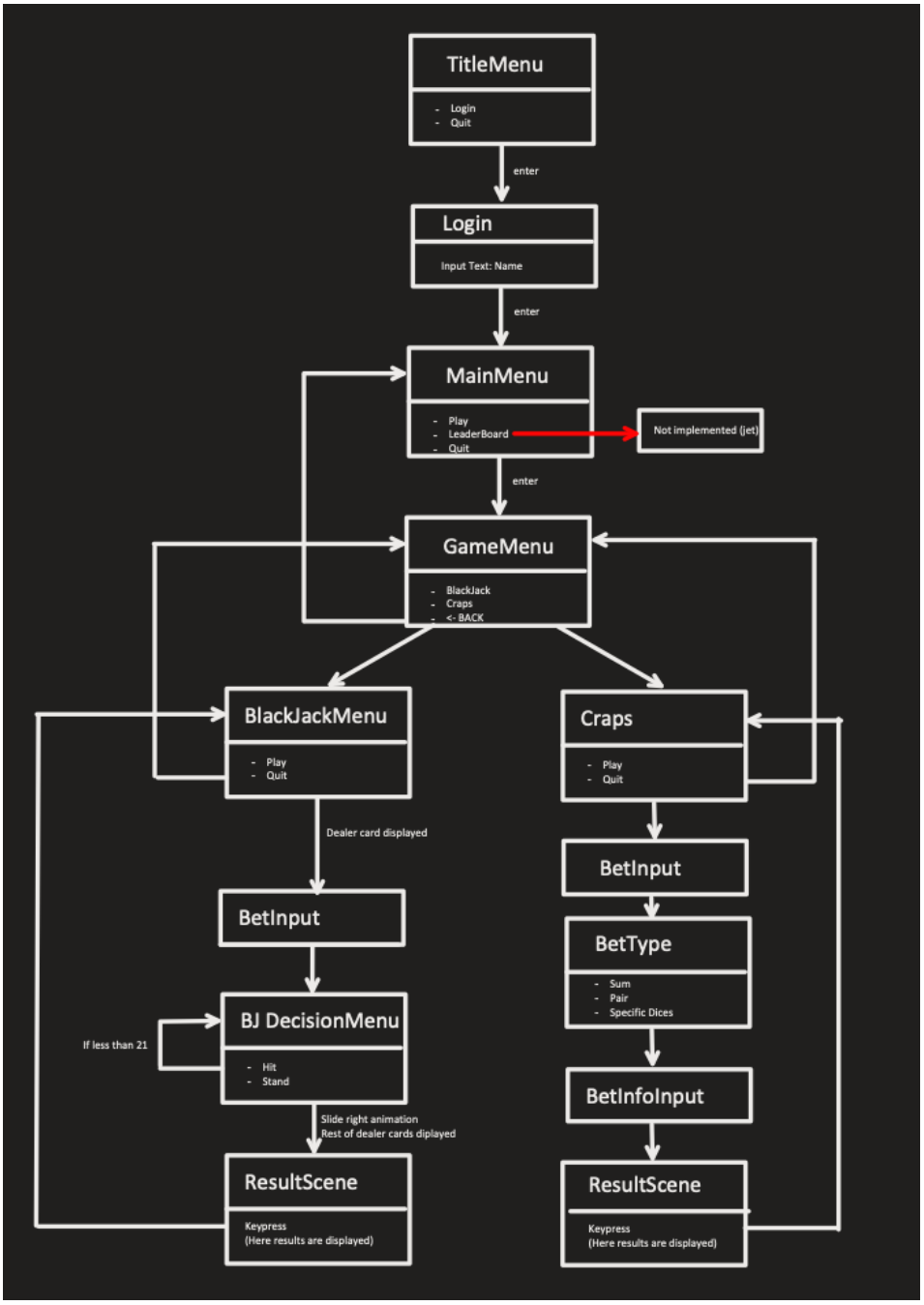
Developers/Authors:

- Adam Jeliński
- Krzysztof Fijałkowski
- Szymon Łukawski

How to play?

```
# 1. clone the repository
git clone https://gitlab-stud.elka.pw.edu.pl/slukawsk/proi-project-casino.o-sl-aj-kf/
cd proi-project-casino.o-sl-aj-kf
# 2. build the application
cmake build
# 3. run the application
cd build
./main
```

Application state graph



Data:

All data is kept in two files - LeaderBoard.csv and Players.csv

- Leaderboard
    - field names: playerId, gameId, score
- Players
    - field names: id, name, password, cash

Description:

The main focus of the development was to make a system that is easily expandable with new functionalities. That's why most of the development work was put into the framework, rather than refined games.

The lifecycle of the application is based around the main event loop, controlled by the main manager. Its frequency is maintained by the timing manager.

What is being shown on the screen is controlled by a simple controller stack. Controllers are fundamental building blocks of the application. During each tick only the controller at the top of the stack receives new events, while all of the other controllers are suspended. This allows us to reuse the same selection and text input menus in the main menu, as well as in games.

During the execution of each event tick a couple things happen:

- The input manager checks if a new keypress has been detected
- If there was a new keypress, the keypress handler of the current controller is executed
- The tick method of the current controller is executed
- The graphics manager redraws the current frame
- The postTick method of the current controller is executed, to allow controllers more freedom (currently not used by any controller)
- The timing manager waits an appropriate time to maintain the correct framerate
- Optionally, the timing manager might draw timing stats next to the main buffer

If a controller wants to end its operation, it commits suicide (removes itself from the controller stack). The event loop is stopped once all of the controllers are destroyed.

The event system combined with menu controllers allow for easy implementation of both turn based (Black Jack and Craps) as well as real time games (Snake) without any modifications to the framework.

The project was built, tested and is working on 3 platforms (Linux, Windows, and macOS).

Main class - main manager:

Holds all of the managers and manages the lifecycle of the application.

Managers:

- Leaderboard Manager
  - is in charge of file with leaderboard, can add score, and return a vector of best 3 scores in game with given id
- Graphics manager
  - displays contents of ImageBuffers to the terminal and configures the terminal for optimal displaying
- Timing manager
  - maintains a constant framerate
- Input manager
  - gets keypresses from the terminal
- Menu manager
  - creates and controls the main menu
  - allows the user to log in
  - starts the games
- User manager
  - in charge of the user data; stores usernames, password logins and user ids
- Built-in UI Controllers
  - SelectionMenu - list of options which can be selected with arrow keys
  - TextInputMenu - text field with support for moving the cursor, as well as the backspace and delete keys

Games:

To visualize our project we made three sample games: Craps, BlackJack and Snake. In all games there is one crucial method - tick. Its invocation tells a game to update its internal “state” and consequently, what is currently displayed. If the game wants to display an animation, the game holds an additional attribute - the frame counter. It counts the number of ticks spent in a given “state” allowing us to dynamically change what is being shown.

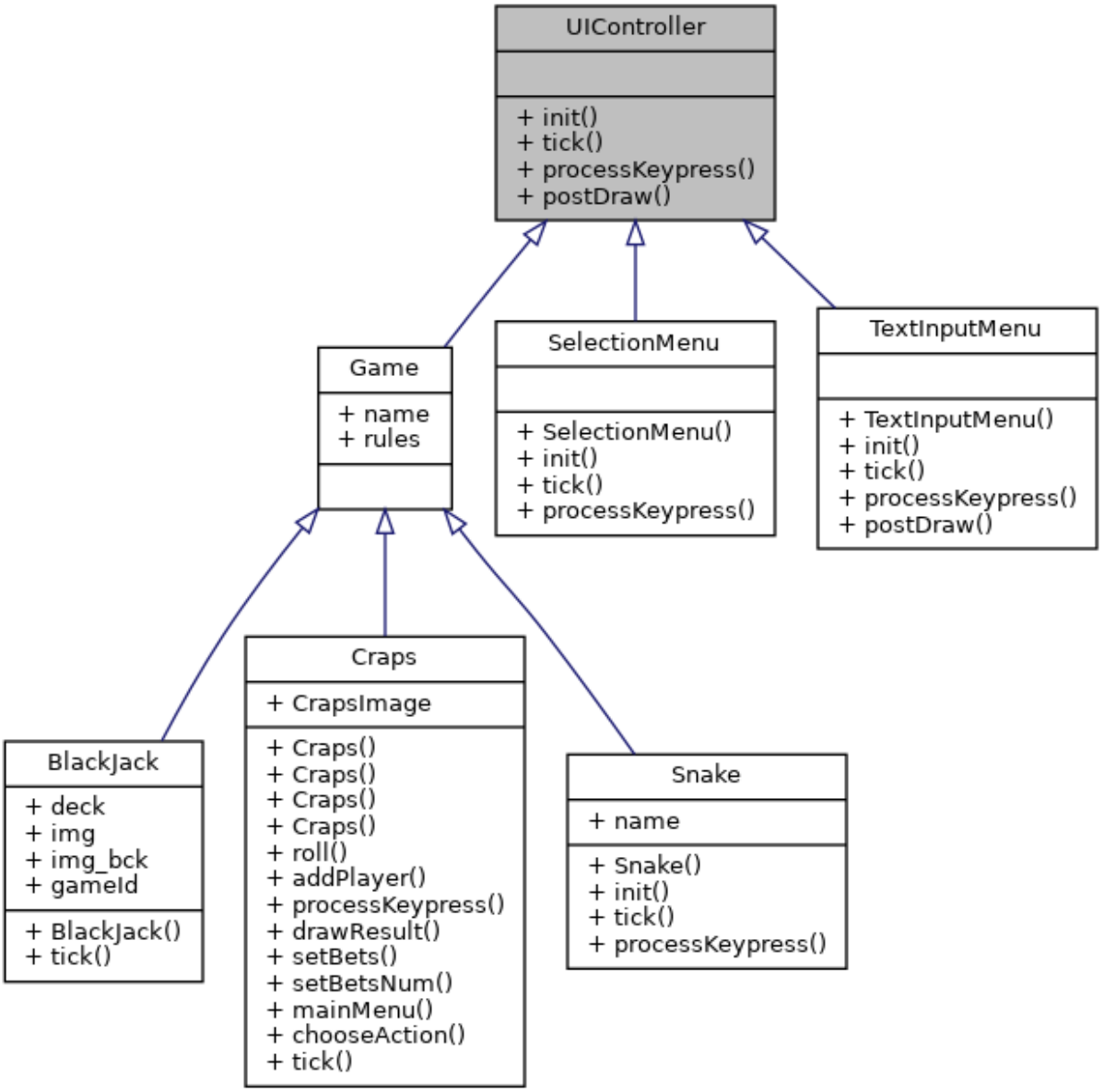
- Craps
  - has dices -> object holding dice object
  - draws dices and menu
  - is initialized by pointer to player
    - uses this pointer to save the current cash state
  - can be easily changed for future multiplayer version
- BlackJack
  - uses Deck, deck uses Card, (it is easier to add new Card game)
  - Card is displayed using ASCII chars (but can be easily improved to use char_32t to get a more detailed picture with Unicode characters)
  - implements animations (getting a new card, and sliding cards to the right)
  - holds attribute “player”, but could be expanded to multiplayer version

- Snake
    - changes the snake direction on a keypress event
    - moves the snake forward every 4 frames
    - keeps track of snake segments using a deque
    - dynamically colors the snake in a rainbow using a rainbow color generator

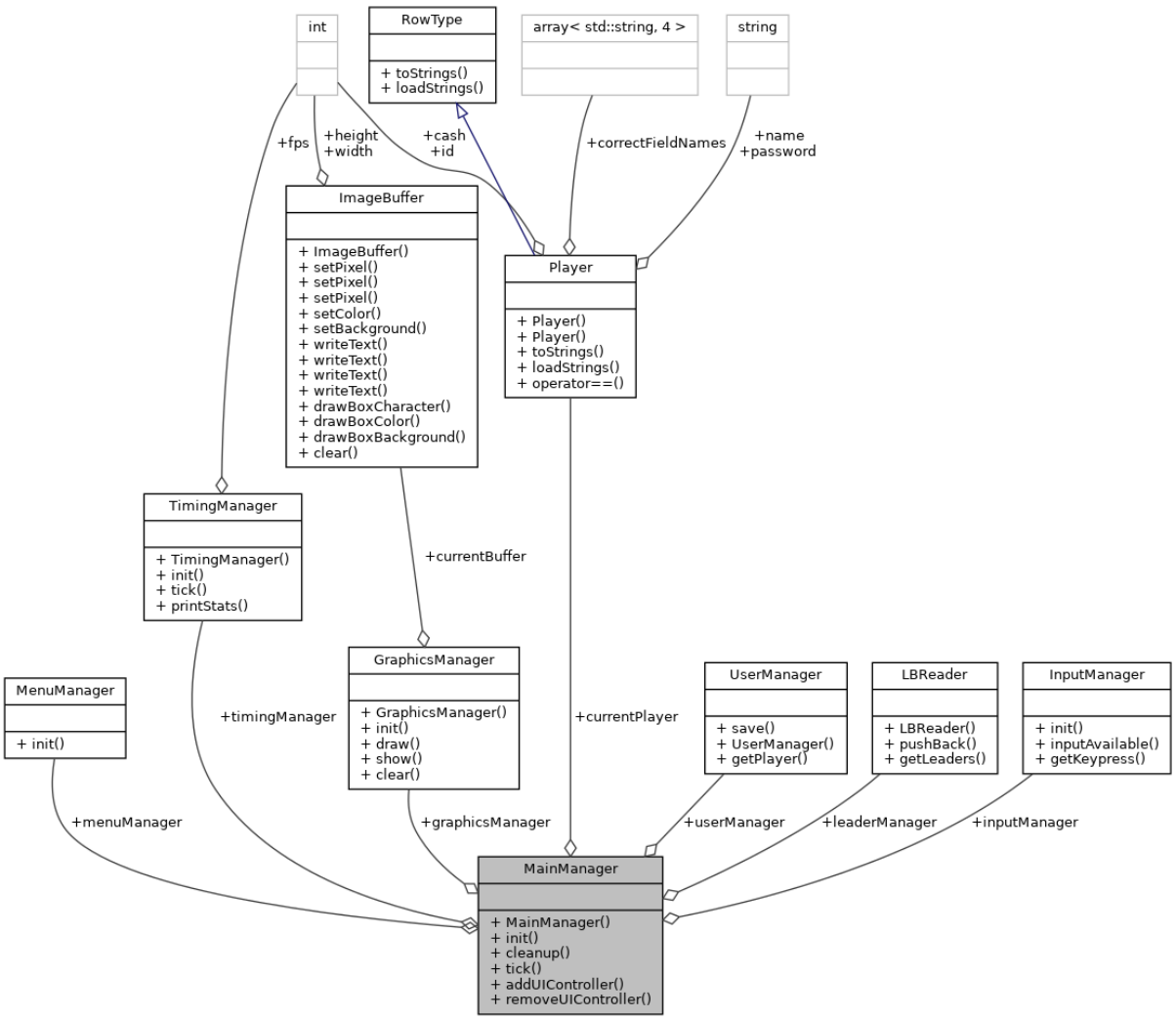
Tasks:

- Adam Jeliński
    - Main manager
    - Graphics manager
    - Input manager
    - Menu manager
    - Both menu controllers
    - Timing manager
    - User manager
    - game Snake
- Krzysztof Fijałkowski
    - Leaderboard manager
    - game Craps
- Szymon Łukawski
    - CSV file manager
    - game BlackJack

Inheritance of UI Controllers

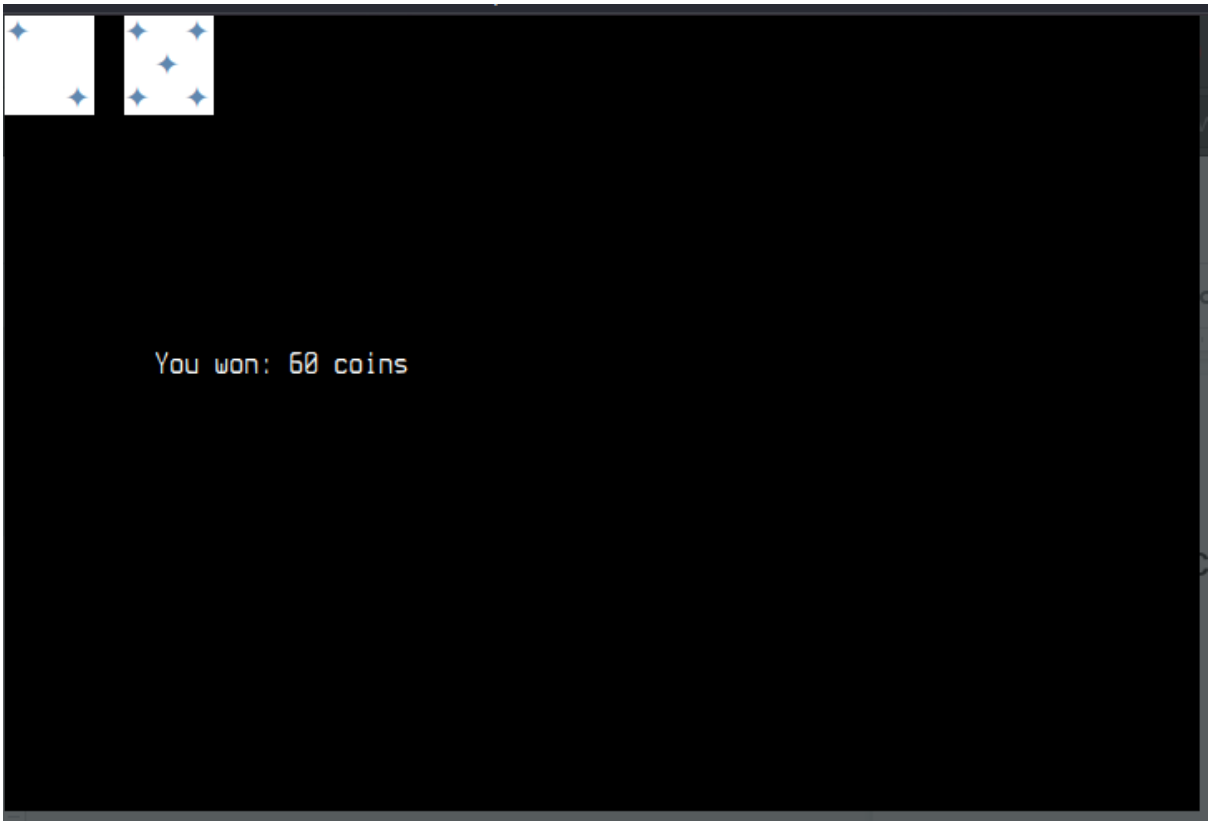


MainManger composition



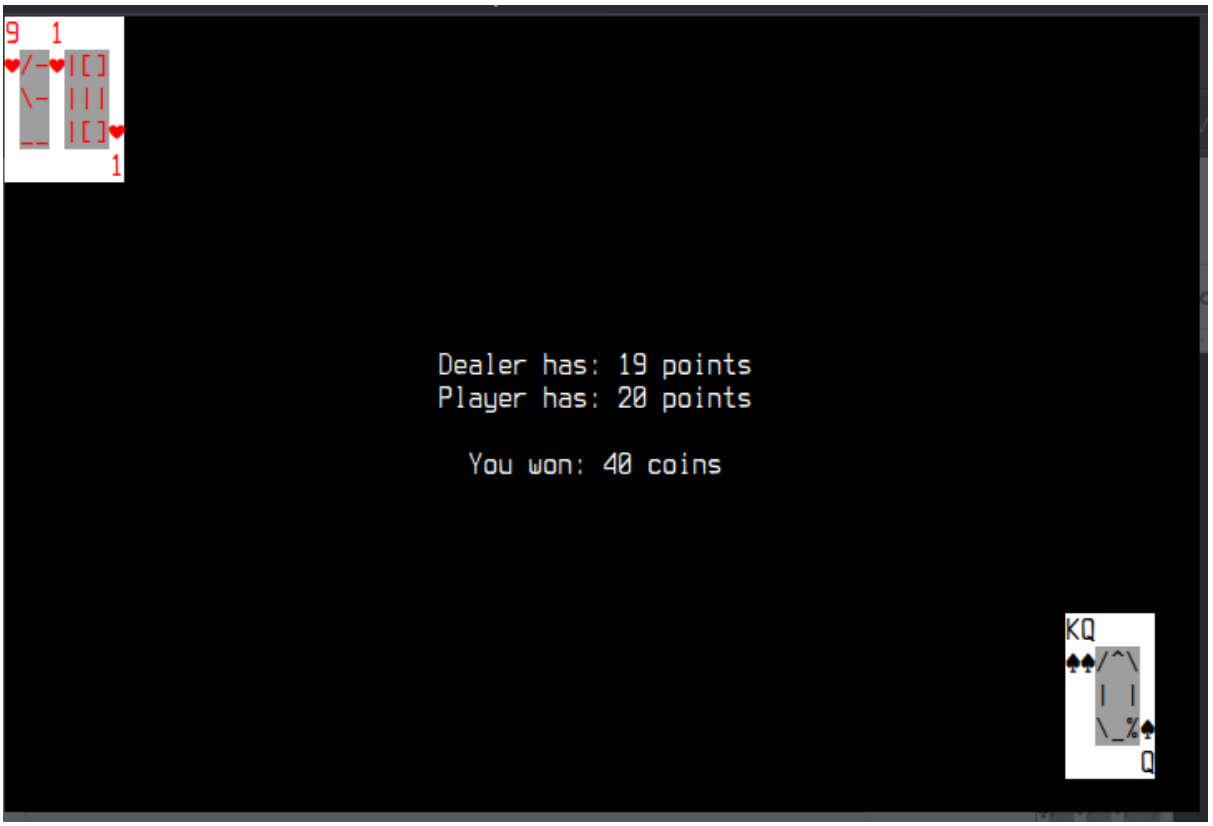
Game screenshots

Craps



Black Jack


What do you do?
Hit
Stand
Look at cards


Dealer has: 19 points
Player has: 20 points
You won: 40 coins

Snake