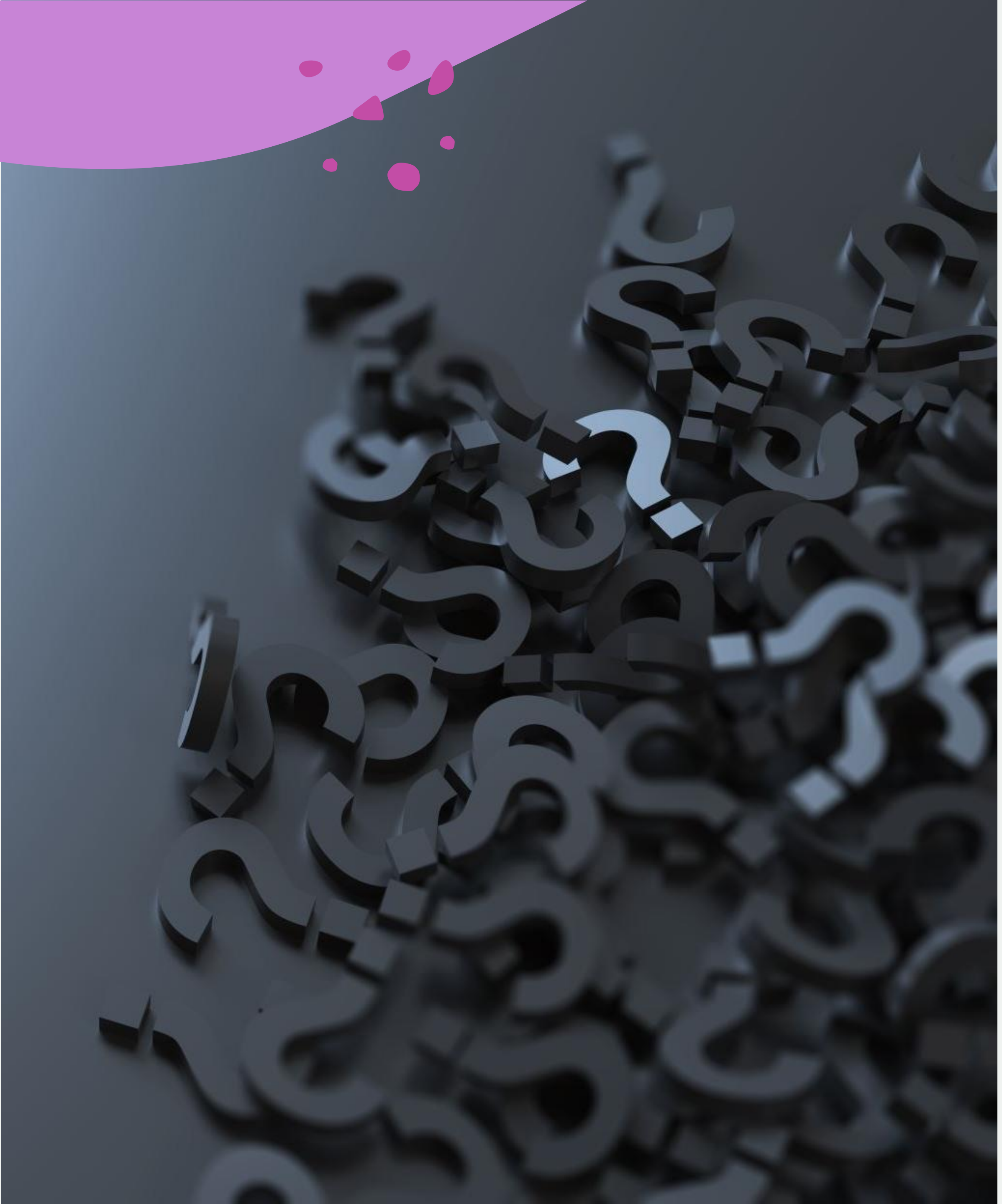


EECS 3311 labs

Week 6: second software project

(with deliverables)

Agenda

- Demo
- Description of the second software project
- Deliverables

General information

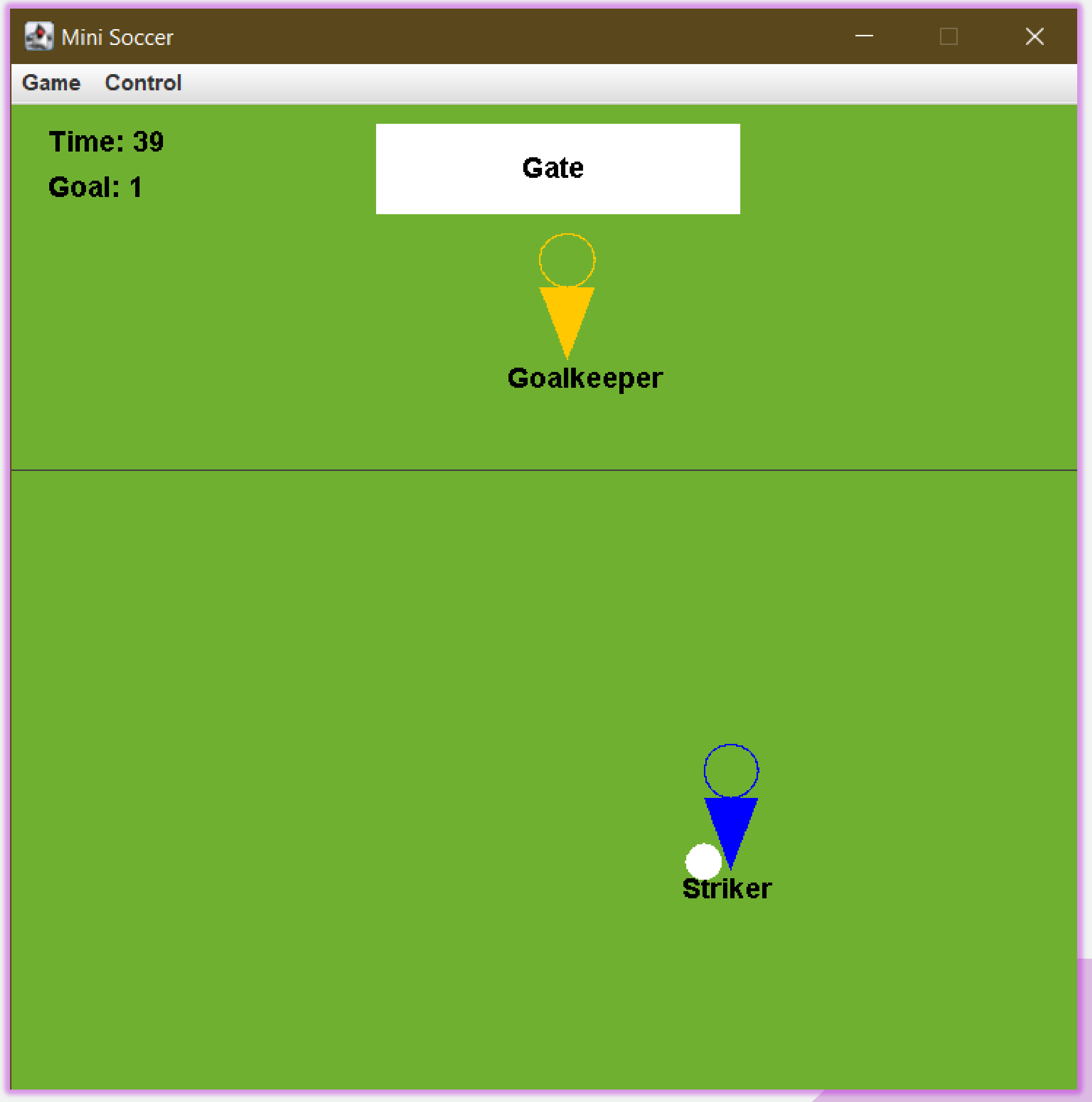
- The software project is worth: **8%**
- You need to complete the software project **in group**
- Deadline of the submission of the project's deliverables: **November 7 (11:59 PM EASTERN time)**
  - **Your group will loose 10% of the grade if you make a late submission!**
- Tools you may use to complete the second software project :
  - Eclipse, Diagrams.Net, Jacoco, GitHub, etc.
- You can work on the software project during the live lab sessions.

Demo of the mini soccer game

Watch the video on eClass entitled ***Lab 5_Mini Soccer game demo*** to better understand how the mini soccer game works.

The mini-soccer game (1/5)

- You should create an application that displays an interface with two menus: Game and Control
  - The Control menu allows pausing or resuming the Game
  - Game menu allows starting a new game or exiting the game
- The interface displays two game players: a striker and a goalkeeper. The goalkeeper is in front of the Gate
- During the game, the striker tries several times to shoot the ball to score a goal and, each time, the goalkeeper attempts to catch the ball.
- The goalkeeper moves randomly on the left or on the right of the gate to try to catch the ball.



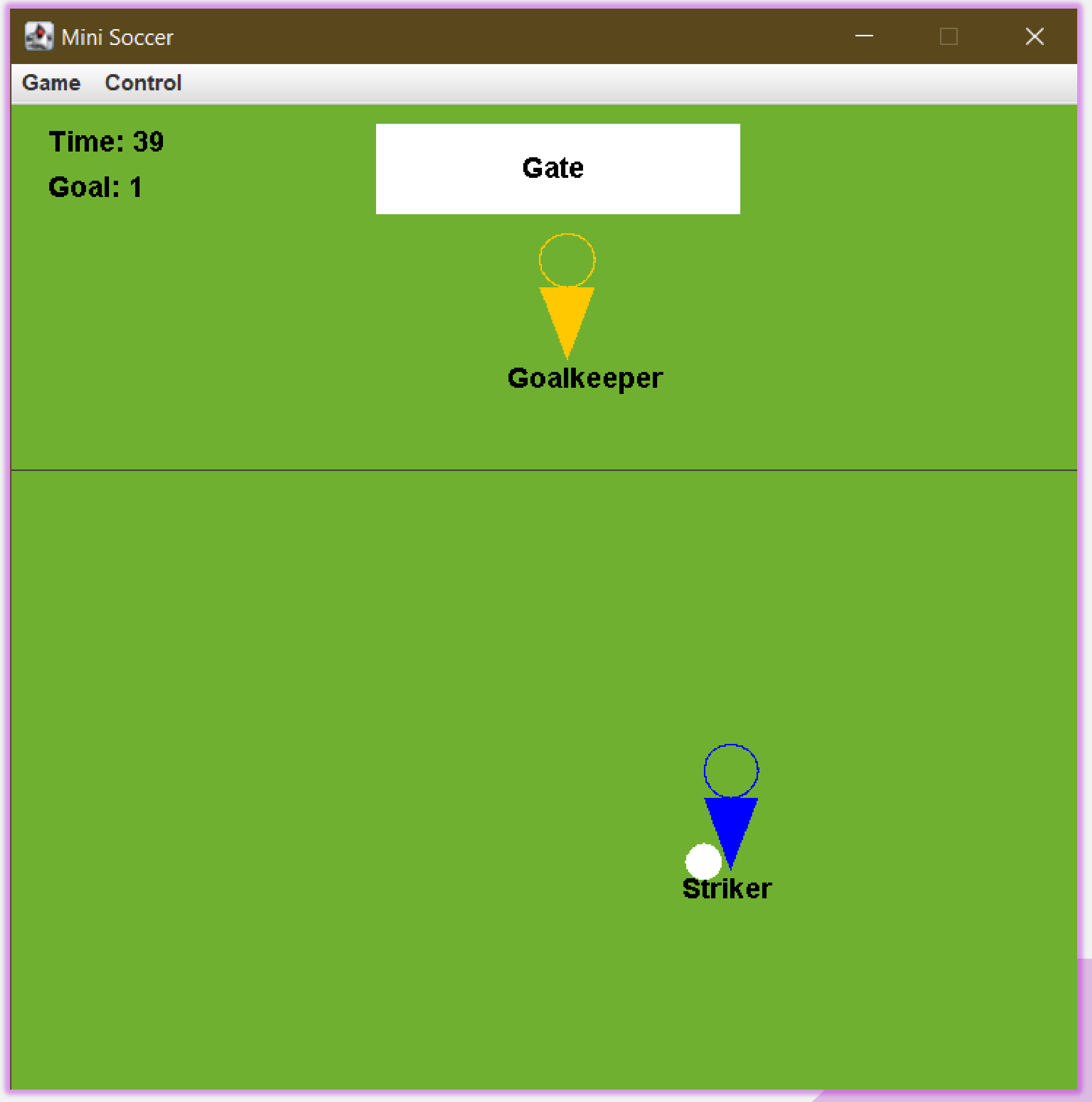
The mini-soccer game (2/5)

The interface should display the **time** remaining time (in seconds) in the game, and the number of goals scored by the striker.

The remaining time starts at 60 seconds. That time decreases during the game. When it reaches 0 second, then the game automatically ends.

You should be able to pause the game, resume the game, and start a new game.

The initial game score is zero.



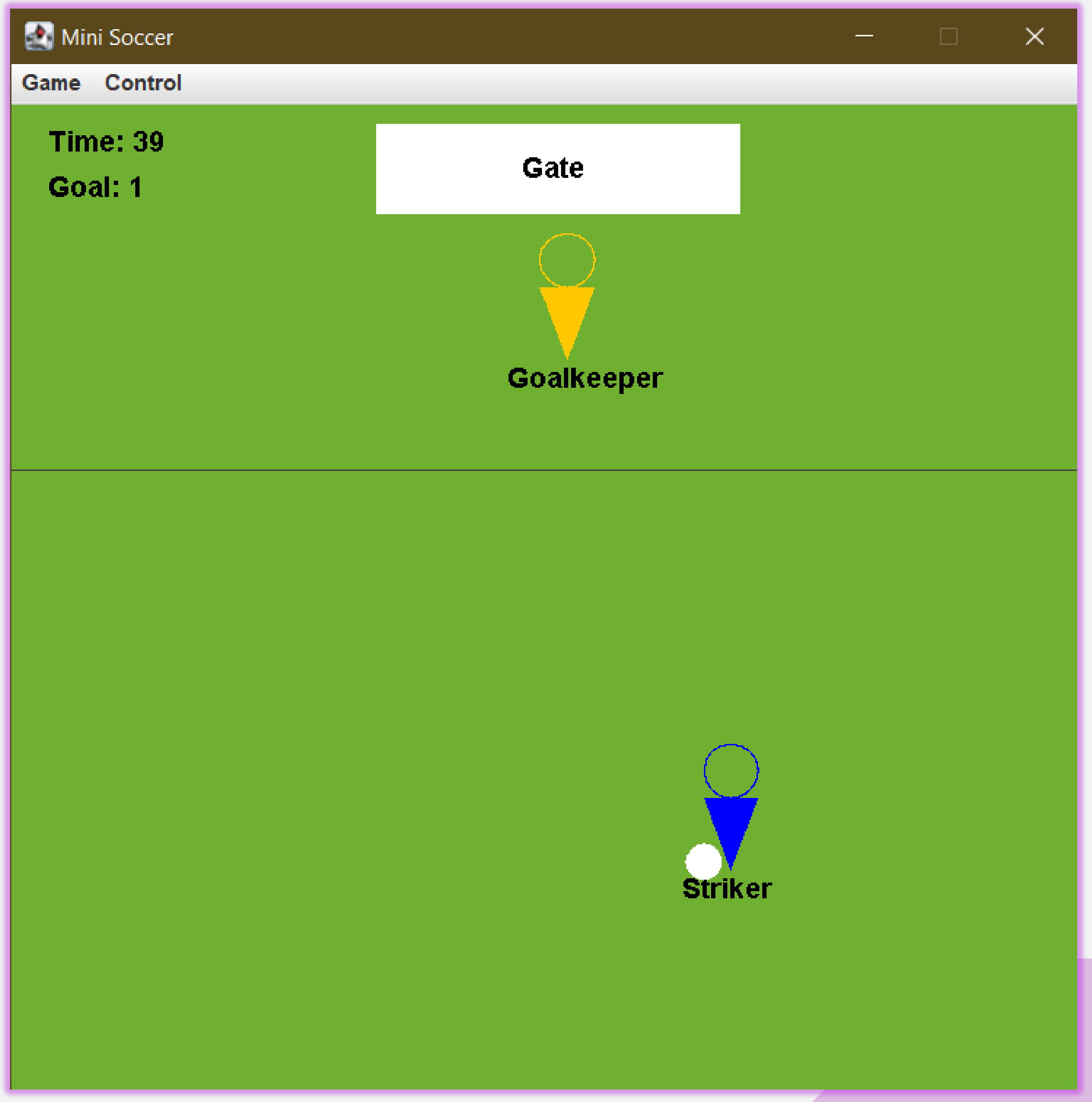
The mini-soccer game (3/5)

Movement rules:

- Use the Arrow Keys to control the movement of the striker.
- The goal keeper moves randomly in a gaussian distribution fashion, where the center of the gate acts as the mean value.
- The striker and the goal keeper cannot cross the penalty line.

Ball control rules:

- Use the space bar to make the striker shoot the ball
- The goal keeper will pick up the ball if the ball falls within its side of the penalty line and not in the gate.
- GamePlayer will pick up the soccer ball if they're within 55 radius of each other.

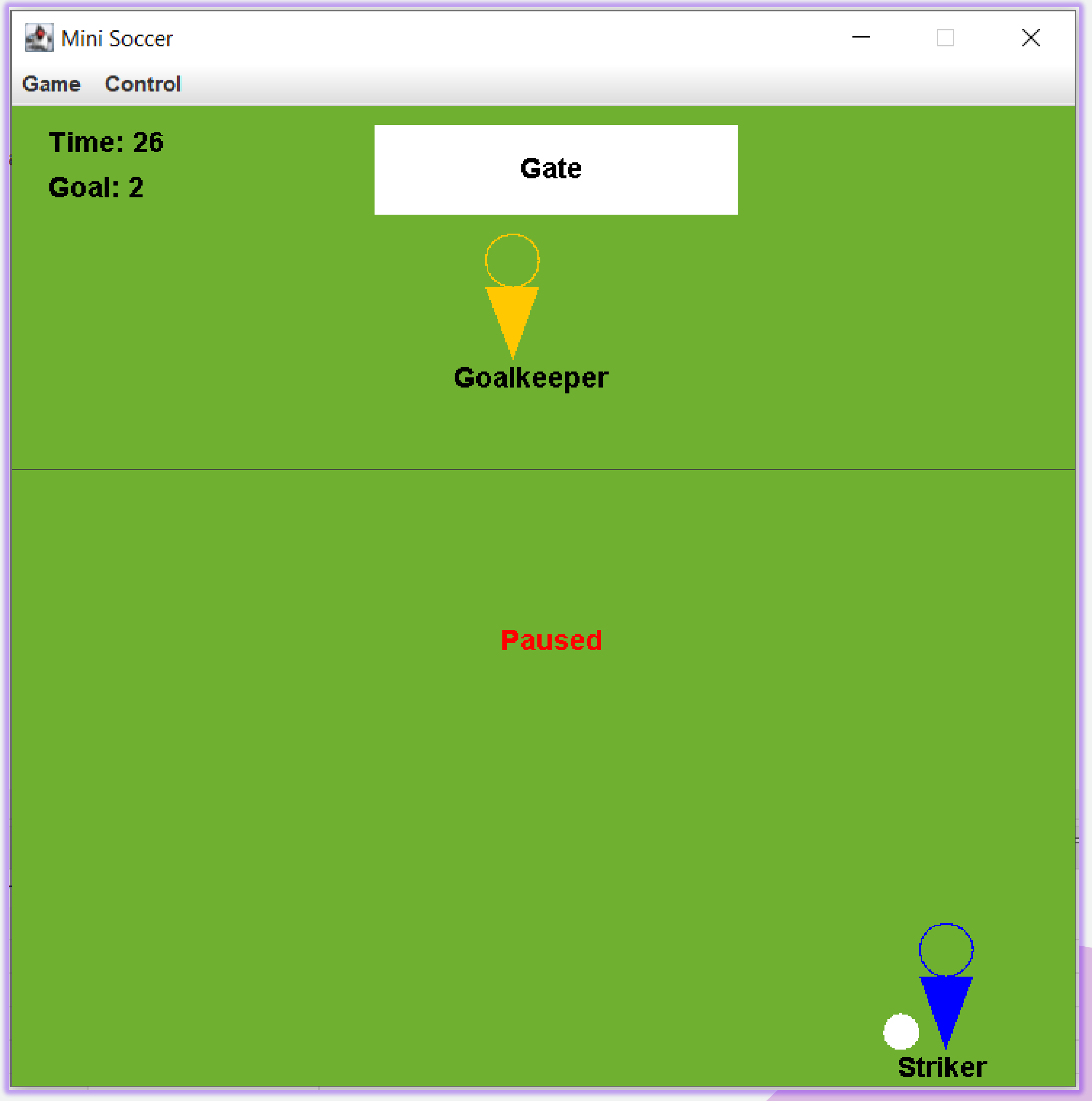


Time: 39
Goal: 1

The mini-soccer game (4/5)

Score/Catch ball rules:

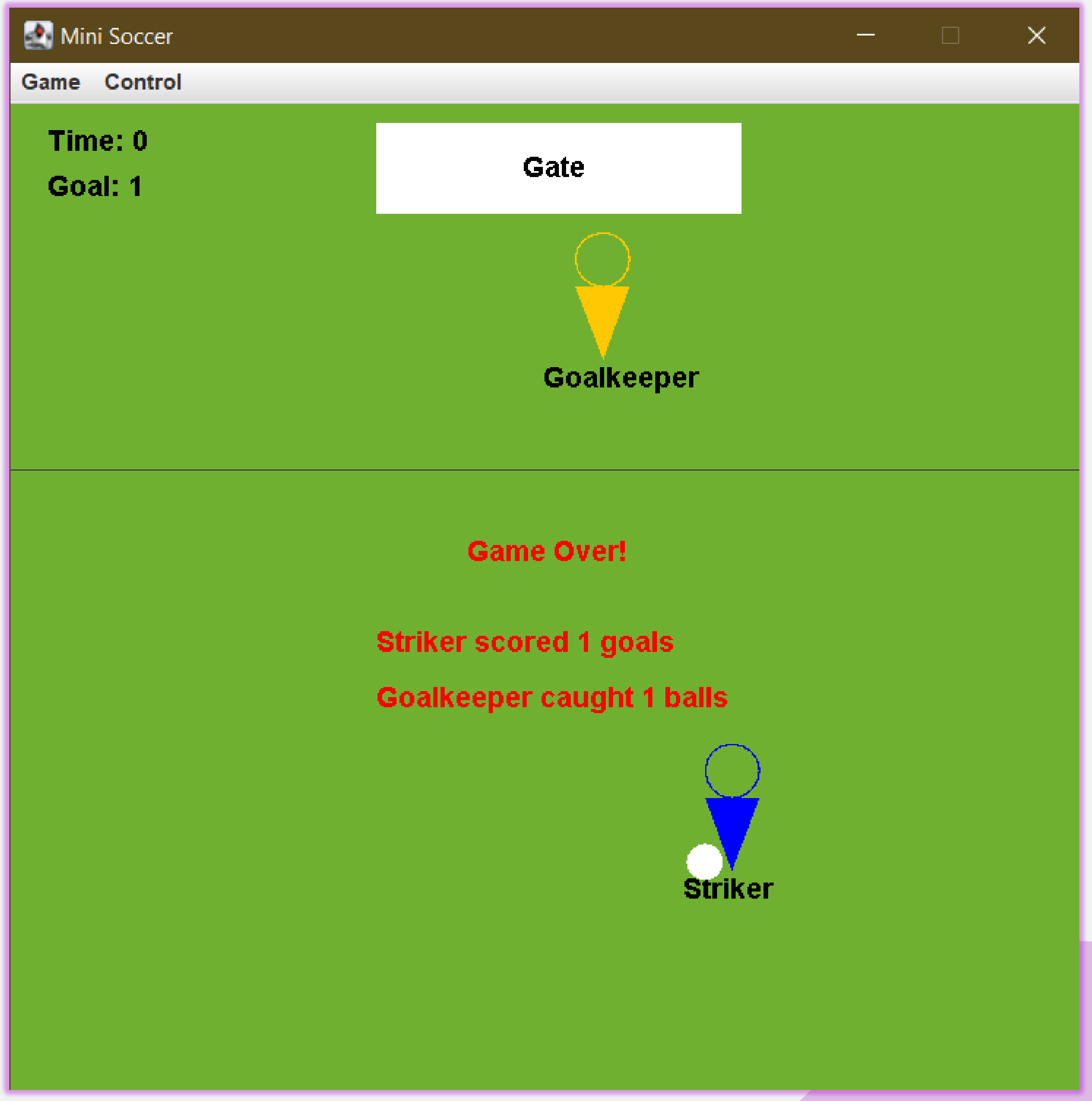
- If the ball shot by the striker lands in the gate area, the goal of the current game will increment by 1, game will set to **pause**, and the position of ball, striker and goal keeper will be reset.
- If the ball shot by the striker lands in its side of the penalty line, nothing will happen.
- If the soccer ball shot by the striker lands in the other side of the penalty line, and not in the gate area, the goal keeper will catch the soccer ball automatically and is considered to have caught 1 ball.
  - ❖ The goal keeper will then kick the SoccerBall back to the Striker side of the penalty line.



The mini-soccer game (5/5)

Game states:

- Game is over if and only if the remaining time is 0.
- If the game is on pause, neither the game player nor the soccer ball can move.
- If the game is over, the statistics of each game player will be sorted and displayed, while all controls, including pause and resume will be disabled until a new soccer game is started.



Game Over!

Striker scored 1 goals

Goalkeeper caught 1 balls

Deliverables: your report (1/2)

The report should be written with a Times Roman (font size =12) and should comprise the following sections:

- A header indicating, for each group member, the name, the student ID, the course, the section, the name of the software project, the name of the TA
- PART I (introduction): up to 1 page, **worth 10 pts**
- Part II (design): up to 3 pages, **worth 45 pts**
- Part III (implementation): up to 3 pages, **worth 35 pts**
- Part IV (conclusion): up to 2 pages, **worth 10 pts.**

Deliverables: your source code (2/2)

- The Source code of your project that should be hosted on GitHub
  - ❖ The code should be able to run without triggering exceptions
- A short video (3 to 5 mins) showing how to launch your code and run it
  - ❖ **Turn your camera off** when preparing the video using Zoom or another capture tool.

PART I: Introduction (10 pts)

- Explain what the software project about and what are its goals: **2 pts**
- Explain the challenges associated to the software project: **2 pts**
- Explain the concepts (e.g., OOD, OOD principles, design patterns) you will use to carry out the software project : **4pts**
- Explain how you are going to structure you report accordingly: **2pts**.

Part II: Design of the solution (45 pts)

- Create a first UML class diagram of your system (***use at least two design patterns***), add the corresponding figure in the report and comment its elements: **(25 pts)**
- Explain in your report how you have used design patterns: name the corresponding classes, interfaces, and if possible most relevant operations **(10pts)**
- Use OO design principles in your class diagram **(10pts):**
  - ❖ Explain in your report how you have used them: name the corresponding classes, interfaces, and if possible most relevant operations

PART II: class diagram main elements

You need to represent all the classes of the MVC in your UML class diagram.

Use the code that is inside the ***MiniSoccerGameProject.rar*** to determine the classes that should be put in the model, the view and the controller.

The model provided in that zipped file is incomplete!

You need to add additional classes in the model so that it includes the following classes:

- **GamePlayer**: any player of the soccer game (e.g., striker, goal keeper)
- **Goalkeeper** : it comprises the goal keeper's attributes and operations
- **Striker**: it represents the striker's attributes and operations
- **PlayerStatistics**: it indicates the statistics of a player (i.e. caught balls, or scored goals)
- **PlayerFactory**: factory that creates players
- **PlayerCollection**: an iterable Collection of players that needs to have methods to initialize a collection of players, to add a player and to remove a player from the collection
- **PlayerCollectionIterator**: iterator that allows iterating over the PlayerCollection.

Part III: Implementation of the solution (35%)

- Complete the code in the ***MiniSoccerGameProject*** to provide the implementation of the missing model classes: **12 pts**
- Describe how you have implemented and compiled all the classes of your class diagram in Java : **6 pts**
  - ❖ The code should be able to run without triggering exceptions
- Write a precise documentation of all the classes (e.g., their methods, and global variables) to explain what each class does. Generate the corresponding Javadoc. **5 pts**
- Write JUnit tests for the model: use Jacoco to reach a 100% coverage. Discuss that in your report: **5%**
- Specify the tools/libraries you have used during the implementation (e.g., Eclipse, JUnit, JDK): **2 pts**
- Create a short video (e.g., 3 to 5 mins) explaining how to launch your application and run it: **5 pts**
  - ❖ **Turn your camera off when recording the video!**

Implementation of the solution

On eClass, you will find the ***MiniSoccerGameProject.rar*** that you can use as a basis to implement your solution. The code inside ***MiniSoccerGameProject*** comprises four packages: model, view, controller, and main. The model is incomplete.

Your code should include four packages: model, view, controller, and main.

- The main package comprise the Java class that allows launching the application
- Your code should complete the code of the ***MiniSoccerGameProject*** to provide the implementation of the missing model classes.

PART IV: Conclusion (10 pts)

Explain the following in your conclusion:

- What went well in the software project? **2 pts**
- What went wrong in the software project? **2 pts**
- What have you learned from the software project? **2 pts**
- What are the advantages and drawbacks of completing the lab in group? **2 pts**
- What are your top three recommendations to ease the completion of the software project ? **1 pt**
- Add a paragraph (or a table) to indicate the different tasks of the work that were assigned to each group member, and the portion of the work completed by each group member. Also indicate if each group member was collaborative. **1 pt**

Deliverables

Create a repository on GitHub to host your deliverables:

- Progressively put your code, your report and all the relevant files in that repository
- To submit your completed work, just invite the TA who hosts the live lab session as a collaborator to your repository:
  - ❖ The TA will have access to your GitHub Project (i.e., the deliverables)
  - ❖ You can learn how to invite a collaborator on a GitHub repository here: https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-github-user-account/managing-access-to-your-personal-repositories/inviting-collaborators-to-a-personal-repository
  - ❖ Use this address to invite the TA: **eecs3311tafall2021@gmail.com**

Lab sessions

You can work on the second software project during the live lab sessions of Week 6, Week 7, and Week 8.

Live lab sessions are delivered through Zoom meetings.

Zoom link for live Q/A, labs sessions and office hours: https://yorku.zoom.us/j/92610715835?pwd=T3FwVTh6Szg1TkpkRmhpVmhodGpwQT09 .

Do not miss them!