Games Development I

Development of a PONG-AI

Introduction
Probably all of you know PONG. The simple arcade tennis game was not the first, but the first commercially successful video game. The game is pretty simple; two players can move a paddle up and down to return a virtual tennis ball and finally score. Joakim made a clone written in Java, the source code is available on studentportalen.

In our version of the PONG game, one player is controlled by a human, using the arrow keys to move its paddle (up and down). The second player is controlled by an AI. The user can select one of three difficulty levels as well as the final score. Selections are confirmed with the Return key, you can go back to the menu using Escape.

Assignment
You work in teams of two persons, which you arrange yourselves. Your task will be to implement three AI classes with different “skills”. The description below will give you all information about the source code that you will need to implement your own AI. Consider the different variables that you can read and set, which are needed to create a clever AI that might even predict the players/balls movement? It should be possible to feel a difference between the difficulty levels and your AI should feel realistic (e.g. the paddle should not jump). Is it hard to implement a “good” AI? How would you define a “good” AI? Consider these questions, we might discuss them during the lecture on the 19th of June. The 18th is also deadline for your results, submit them via studentportalen. You should not spend more than four hours on the assignment.

Description of the Source Code
The PONG game is quite simple, thus it does not use the structure given in the course book “Developing Games in Java”, which is useful for more complex game projects.

Classes
Main.java
The main code is inside this class, what makes everything tick and render out on the screen.
PaddleHuman.java/PaddleComputer.java
These are the two paddles on the screen. PaddleHuman is the one on the left and is controlled by the player, while PaddleComputer is controlled by the AI, and is on the right side.

Ball.java
The ball that bounces around on the screen, and adds to a paddles score when reaching the goal-line right outside the screen.

AI.java/AIeasy.java/AIhard.java/AInormal.java
First there is a main AI class, which is abstract and is just used as a base that all the different difficulties extend. The AI that will be steering the opponent paddle is selected on the title screen (easy, normal, and hard). These are the classes you are supposed to implement. You may add as many help-functions and variables as you want inside these classes. However, the only thing you really have to implement is the tick();-function, where you can control the movement of the AI-controlled paddle. The returned value will set the y-position of the AI-controlled paddle.

Note that the paddle does have some boundaries. E.g., it cannot go outside of the screen, so if you return a value which would place the paddle outside of the screen (For example -256), it would move to the closest valid position (in this case 32). The actual boundaries for the paddle are a minimum y-position of 32, and a maximum of 352.

Variables
Interesting variables are made static, so you can easily refer to them using the name of the class, and you do not need a specific instance of that class.

These variables can be accessed:

- Ball.getX();
- Ball.getY();
- Ball.getWidth();
- Ball.getHeight();
- Ball.getVelocityX();
- Ball.getVelocityY();

The information about the ball is probably most important to you. You can use the actual position of the ball as well as its velocity (this will determine the balls movement) in order to predict its movement. The function bounceAgainstObject(...) in the Ball-class calculates and executes the change of velocity of the ball after it got hit by a paddle. You might use its content to calculate in advance, where the ball will go depending on where you hit it. When the ball hits a wall, only its vertical velocity will get multiplied with -1.
• PaddleHuman.getX();
• PaddleHuman.getY();
• PaddleHuman.getWidth();
• PaddleHuman.getHeight();
• PaddleHuman.getVelocityY();

Information about the paddle controlled by the human player might help your AI to decide, where to aim the ball to.

• PaddleComputer.getX();
• PaddleComputer.getY();
• PaddleComputer.getWidth();
• PaddleComputer.getHeight();

• Main.getScoreHuman();
• Main.getScoreComputer();
• Main.getScoreLimit();
• Main.getFieldWidth();
• Main.getFieldHeight();

You might need these variables to pre-calculate the balls movement. Using the actual score might even give you a simple possibility to give your AI a character or to implement an AI with adjusting level of difficulty.

Additional information
• The top-left corner on the screen is the position (0, 0); the value on the X-axis increases downwards, the value on the Y-axis increases into right direction.

• For the paddles and ball objects, the X and Y positions do not point towards the objects center, but rather there top-left corner.

• The "scoreLimit" is the score at which the match will end.

• The velocity of an object is the current speed it has in a certain direction. The paddles can only move on the Y-axis.

• For each bounce it does, the ball will slightly increase its speed up to a maximum.
Things you might want to think about

- It is fairly obvious how the AI can play defensively. Can it play offensively? How?

- When getting X and Y value of an object, we point towards the top-left pixel. How do we get the bottom-right pixel, or the center?

- The velocity values of the computer controlled paddle do not exist. Can we create these in the AI classes? What could we use it for?

- The ball bounces differently depending on the angle between the balls center point and the paddles center point. Can we calculate the angle at which the ball will bounce when it hits the computer controlled paddle? Can we use this?

- What is the worst AI you can imagine, in this case?