School of Computer Science
University of St Andrews
2018-19
CS4303
Video Games
Practical 1: Particle Artillery

This Practical comprises 20% of CS4303. It is due on Wednesday 3rd October at 21:00.

The deliverables consist of:

- A report.
- The Processing source code for the video game you will write.

**Background**

This practical is intended to give you the opportunity to learn the Processing language, and implement some of the concepts from the Physics component of the module.

The task is to implement **in Processing** a variant of the classic Artillery Game. If you are not familiar with this type of game, your first task is to read the summary at Wikipedia:

https://en.wikipedia.org/wiki/Artillery_game

The Atari 2600 game Artillery Duel is close (but not identical, see below) to the specification for this practical. You can see it in action on YouTube:

https://www.youtube.com/watch?v=NsopDPcF7co

**Problem Specification**

Particle Artillery is a two-player game in which each player controls a tank. The players take turns to fire shots at each other, with a point scored for a hit on the opposing player (if a player unfortunately hits herself, then the point goes to the other player). The game ends when one player's score reaches some predefined limit, whereupon that player is the winner.

Elements for you to decide (and document):

- What is the score limit?

**The Play Area**

The play area is a two-dimensional space with ground level near the bottom. The two tanks are at ground level and at opposite ends of this space, facing each other. Between them is a landscape composed of rectangular blocks. The play area is subject to gravity, and wind, which may change direction between turns.

Elements for you to decide (and document):

- The dimensions and look of the play area, landscape blocks, and tanks.
- How is the landscape generated? Is it always the same?
- The strength of the force of gravity and wind relative to the dimensions that you have chosen.
- How often the wind can change, and how strong it is.

**Controlling the Tanks**

The player can drive her tank forwards and backwards, but the tank cannot drive through a block, off the edge of the play area, or through the other tank. A simple implementation of movement by directly modifying the tank's velocity, and of these collisions, resolved by the tank simply coming to a dead stop next to the obstacle, is acceptable.

The player also controls elevation (a normalised vector indicating the direction of fire) and strength of shot (the magnitude of that vector). Having set these controls, the player can fire a single shot, ending her turn.

Elements for you to decide (and document):

- The implementation of the controls: mouse, keyboard, … ?
- Collision detection for the tank (when driving).

**The Shell**

Firing should produce a force according to the elevation and strength settings on a particle (with an implementation based upon the Particle class given in lectures) representing the shell fired. The shell should also be subject to the force of gravity, the force of friction due to movement in the air (which can be modelled via a damping factor, as per lectures), and the wind present. These forces should be **accumulated** as shown in lectures.

If the shell collides with the ground, it is destroyed and the turn ends. For simplicity, the ground is unaffected. If the shell collides with a landscape block, both the shell and that block are destroyed, and the turn ends. NB In the basic specification you are **not** required to implement gravity affecting the blocks – even if a block has no support beneath it, it should not move. If the shell collides with a tank, it is destroyed, a point is scored as described above, and the turn ends. You are also **not** required to model any explosive forces from the collision of the shell with the ground, blocks, or tanks.

Elements for you to decide (and document):

- The dimensions and look of the shell.
- Collision detection for the shell.

**User Interface**

The following information should be displayed:

- The score of each player.
- The elevation and strength currently selected by each player.
- Whose turn it is.
- The wind strength and direction

Elements for you to decide (and document):

- Will you have an introductory screen giving instructions before the game starts?
- When the game ends will you provide the option of restarting the game?

## AI Player

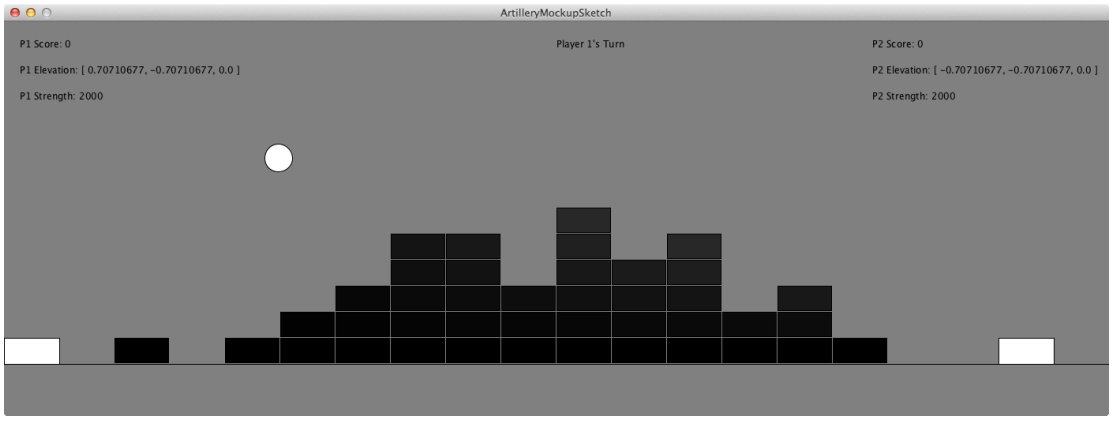
Finally, add to your game the option to play against an AI opponent rather than another human player. This option should be available via a menu or similar before the game begins.

Elements for you to decide (and document):

- How will your AI decide shot elevation and strength? Consider the balance between making the AI too good or too bad at playing the game.
- How will your AI decide when and where to move?

## Mockup Seen in Lectures

We saw and discussed a simple mockup of this game in lectures. A screen capture from this mockup is presented below for your reference:




## Report

Your report should document the design and implementation of your game. Include screenshots that show your game in operation and illustrate its features. Your report should include discussion of all of the design decision points mentioned in the specification above (plus any extensions), as well as an account of your implementation of the physics involved.

For at least one element of your game, compare how different design choices affect the game for the players. Justify the choice you have made for the submitted version of your game.

## Marking

The practical will be marked following the standard mark descriptors as given in the Student Handbook (see link below). There follows further guidance as to what is expected:

- To achieve a mark of 7 or higher: a bare bones implementation of the game, allowing each of two players to control a stationary tank and shoot a shell at the opposing player by selecting strength and elevation as described above. The shell need only be subject to the force of gravity. This implementation should be adequately described in an accompanying report.
- To achieve a mark of 11 or higher: In addition to the above, the landscape of blocks between the two players must be added to the game, with collision detection between the shell and blocks implemented. Blocks do not have to be destroyed when hit by the shell. The player should be able to drive the tank forwards and backwards as specified, again with collision detection preventing driving through blocks or off the play area. This implementation should be well described in an accompanying report.
- To achieve a mark of 14 or higher: In addition to the above, blocks should be destroyed when hit by the shell, and the shell should be affected by friction and wind. This implementation should be well described in an accompanying report.
- To achieve a mark of 17: the full basic specification above must be implemented and the report written to a high standard.

## Extensions

There follows a list of possible extensions to your game. These are not required to gain a mark of 17, but at least one extension item (either from this list or of your own design) must be well implemented and documented to gain a mark above 17.

- Can you add bonus items to the play area to aim at/score extra points? These could be special landscape blocks, or they could perhaps appear in the sky.
- Can you make gravity affect the blocks, so that removing the support beneath a block makes it fall?
- Add sound effects! See the minim library for Processing.

## Pointers

Your attention is drawn to the following:

- Mark Descriptors:

  `https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html`
- Lateness:

  `https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html`
- Good Academic Practice:

  `https://info.cs.st-andrews.ac.uk/student-handbook/academic/gap.html`