CS5041 – Practical 1 – Phidgets

Alex Wilton
School of Computer Science
University of St Andrews
ajw28@st-andrews.ac.uk

ABSTRACT
The aim of the practical was to gain practical experience in the area of physical computing and programmable interfaces that can modify the physical world beyond a screen. A number of Phidget components were used to create a controller interface for the Axteroids program, which was adapted to be fully controlled using Phidget inputs. The original program was extended to incorporate a spaceship laser and a moving enemy ship, which can be shot at.

INTERFACE COMPONENTS
The interface for the Axteroids program was created through the combination of 5 types of Phidget components.

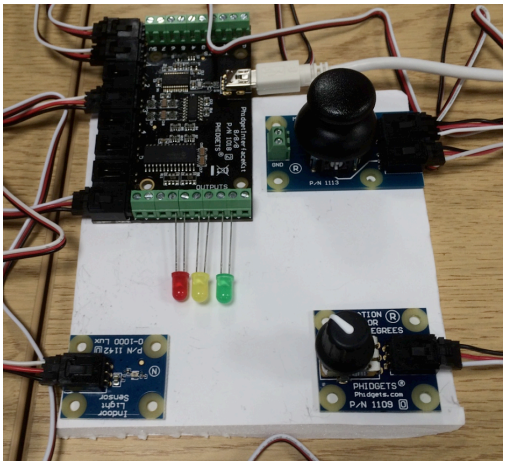

Figure 1. Controller for Axteroids program comprised of Phidget components.

The follows table explains what each component is and how it was used to interface with the Axteroids program:

Component Type	Location in Interface	Program Use
Light Sensor	Bottom-Left	Fires laser from Spaceship when covered.
LED (Green, Yellow and Red)	Centre-Left	Shows state of laser to player. Green means ready to fire. Yellow means laser recharging. Red means laser currently firing.
Rotational Sensor	Bottom-Right	Sets the speed of the Enemy Ship.
Ministick Sensor	Top-Right	Steer and set acceleration for spaceship.
Phidget Interface Kit	Top-Left	Provides a USB interface for accessing all the sensors.

SOFTWARE DESIGN
The design pattern of Model-View-Controller has been used for the overall program structure. The GameModel instance contains instances representing both the players and the enemy's spaceship. The View is created by displaying visualisations from the game model. The Controller contains methods for interfacing with the Phidget components.

The Game Loop which was called once per frame, told the model to update itself for the given frame, updated the model with information obtained from the controller inputs (e.g. laser fire, changes to spaceship direction, acceleration and enemy speed) and set the controller outputs (e.g. which LEDs to light).

Both spaceships were represented by subclasses of the MovingObject class. They both had direction as well as position, velocity and acceleration vectors. When a spaceship was updated for a given frame, the position was updated using the velocity and velocity is updated using acceleration. Notably, to make a spaceship easier to control, a small amount of drag was applied at each update.

ARTWORK

Figure 2. The enemy spaceship (left) and the player's spaceship (right)

As you can see from figure 2, graphical artwork has been used to depict spaceships. Both images were sourced online (and are under the Creative Commons License). Artwork was used as it is makes the game more aesthetically pleasing for the player than using triangles.

GAME PLAY

The idea behind the game play was very simple. The player must try and get close to the enemy ship, then use the ship's laser to destroy the enemy. However, the enemy ship would try and move away from the player's ship. Difficulty can be adjusted by using the rotational sensor to set the speed of the enemy ship. If the enemy ship was it by a laser, if would blow up and a new enemy ship would reappear 7 seconds later. The spaceship laser would fire for 2 seconds before then requiring 5 seconds to recharge. (LEDs would show the laser state to the player).