CS5041 Practice1: Bricks

Hantao Sun
University of St. Andrews
St. Andrews, the United Kingdom
hs99@st-andrews.ac.uk

ABSTRACT
UPDATED—15 October 2018. This report is a part of delivery of course CS5041 in 2018.

Keywords
Phidget; Processing.

INTRODUCTION
There are kinds of ways for human to interact with computers. While mouse and keyboard are the most common ones, more physical methods would provide users with experience more interesting, make the interaction more efficient. To test a certain technology, games could be regarded as proper samples. This report will introduce a game called Bricks, built in processing 3 and implemented with a Phidget kit, a rotation sensor, a mini stick, an indoor light sensor and three LEDs whose color is red, green, yellow respectively.

MECHANISM
In this game, players are supposed to control a long board to launch the shell and bounce it. The goal is to let shell hit bricks in the top of the screen and get scores. Shell would bounce when it collides with bricks, the board or the border of top, left, right sides of play area.

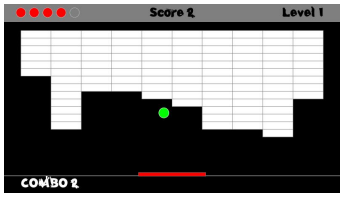

Figure 1. A screen capture during playing

There are some interesting mechanisms in this game. Players have up to five lives. Every time they missed the shell and let it fall onto the ground, they would lose one of them. There are two ways to gain lives, the first one is constantly destroy 50 bricks without losing a life, which is marked as "50 COMBO" and shown at the bottom of the screen. Every 10 COMBO points would also increase the score of a hit of shell. For example, if player destroy a brick while he has gotten 10~20 COMBO, he would get 2 scores, or 3 scores if he has 20~30 COMBO.

Figure 2. (Left) A normal combo mark

(right) A combo mark with an extra life

The other one is to destroy all bricks in the screen, followed by a regenerate of bricks and game would step into next level. Players could check their life, score and level in the information panel at the top of screen.

Figure 3. Information panel

If player lose all lives, they will have to choose to restart the game or exit.

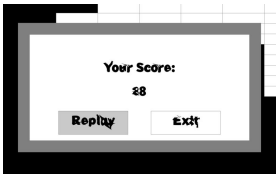

Figure 4. UI after player lose all chances

Some other details of the design and the implement of the game would be indicated in following parts.

INTERFACE
In this part, physical inputs and outputs will be indicated detailly.

There are 5 kinds of operations: adjust of the angle, move the board, launch the shell, switch between two buttons of the dead panel and press the button.

Adjust the angle (rotation sensor)
Turning the rotation sensor, players could adjust the direction of velocity of shell. There is a aim line showing the route of shell before the first bound, to help player decide the direction.

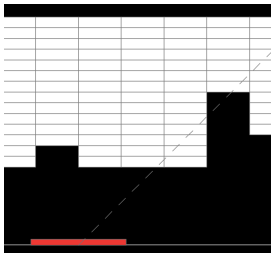

Figure 5. A screen capture during aiming

Move the board (mini stick sensor)
The movement of the board is implemented with the mini stick sensor. Players could rotate the stick horizontally to move the board to the left or right, which may be the most main operation of the game.

Certainly, the board could not move out of the screen.

Launch the shell (indoor light sensor)
Once players finish the prepare phase and be ready for firing, they could launch the shell by grasping the light sensor (or other way keeping the sensor away from light). Although it may be fallible, light sensor is a better choice than temperature sensor and sound sensor to accomplish the fire operation. Although digital input on the mini stick sensor is also a good choice, firing and moving should be separated to two sensors to avoid maloperations.

Digital outputs (LED)
There are three LEDs of different colors utilized in the design of the game. The green LED would keep shining in prepare phase during player adjust the angle of firing and position of the board. As long as they switch to launch the shell, the red LED would be turned on and the green one would be turned off. Finally, the yellow one is used to blinking when players lose a life.

Switch and press button (mini stick sensor)
Just like moving operation, players touch the stick sensor to make decisions after they lose all lives. If they want to confirm their choices, they could press the stick as a digital input. To avoid switching the button when they press the button, the switching operation will be valid only if the value of the sensor is in a proper range.

PARAMETERS

There are some parameters set for design reasons

Screen size
The game is run in full screen mode. Because larger play area could make it easier to recognize various elements in the game and concentrate on playing experience.

Number of bricks
In every level, there would be 10 columns of bricks in the screen. However, the number of bricks of each column is random between 5~15.

Play parameters
The length of board, the speed of shell and board, and the size of bricks are adjusted to a appropriate value.

COLLISION DETECTION OF SHELL

The collision detection of the shell and square items, including board and bricks, is the most complex algorithm utilized in the game. The general thinking is to connect the center point of the circle (shell) and the square with a line. If the crossover point of this line and the circle is in the square area, the collision has happened.

OTHER SOURCES FROM INTERNET

There are some other sources used in game, which are come from the Internet:

Sounds
If the shell collides with the board or a brick, there will be a collision sound [1], and players could hear some laugh when they lose a life [2], which comes from a famous meme in Chinese Internet culture. Sound playing is implemented with Minim library of processing.

Font
The font of text in this game is cited by a TTF file called *u27fog.ttf* which is gotten from the Internet. [3]

Preference of all these sources will be given in later part.

CONCLUSION (TECHNOLOGY COMMENTORY)

Although processing, which is mostly based on java, is not a perfect tool to develop a game, its wild range of graphical functions of both 2D and 3D prove that it could be one of the best starter tools for graphical programming. Drawing frames, which may be the core thinking of processing, make the process of programming quite interesting.

Phidget makes the interaction not limited in mouse and keyboard. Physical signals such as sounds, temperature and light could be efficiently used to “communicate” with computer, which may be the most change of human computer interaction way in recent years. It’s also very friendly to developers, as the thorough documents are provided and there are many versions of API in different languages. However, combined with processing, sometimes it could not detect Phidget devices at first frames, making some functions unavailable (such as that used to set the data interval of the channel), which shows there are possibly something need to be improved.

SOURCES REFERENCES

1. collide.mp3. Available from: http://www.aigei.com/audio/
2. laugh.mp3. Choi Seong-guk. 2007. *Mr. Kim vs Mr.Kim vs Mr.Kim*
3. u27fog.ttf. Available from: http://www.font5.com.cn/