University of Dhaka

Department of Computer Science and Engineering

Project Report:
Fundamentals of Programming Lab (CSE-1211)

Project Name:
Bow and Arrow game

Team Members:
Md. Sadmin Tahmid Khan
Ishwor Dhungana

1. Introduction

Our Bow and Arrow game is based on Archery. The game features an Archer that can be controlled by the user using key presses which includes changing his position and shooting arrows. Several boxes will move upwards at a certain speed. Each type of boxes has a feature. Some boxes will earn the user points. Other boxes has specializations. Those boxes will either make it easier or more difficult for the user to successfully shoot boxes and gain points.

2. Objectives

The objective of the project is to apply our knowledge on coding techniques and programming methods into a totally new platform – SDL. Also, we learnt about different features and tools of SDL, new ways of efficient coding. Our objective was also to combine our prior and new programming skills with those features of SDL that we learnt to create the game of our desire.

3. Project Features

We used buttons in all pages as shown above. Using mouse the user can navigate through all pages by clicking on the buttons.

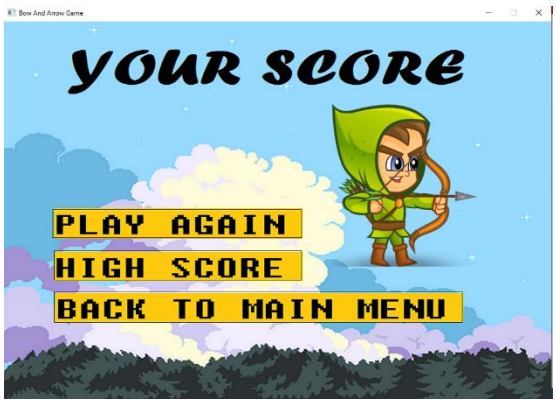
We have a feature that shows the games high score. Also, we have another feature by which the high score can be reset.

The above screenshot is the first page of the instructions. It has a detailed information about some of the features. This includes using ‘up/ down’ arrow keys to navigate the archer’s position and using ‘space’ key to shoot and earn points.

The above screenshot is the second page of the instruction. These include the most important features of the game like changing the size of the boxes, speed and freezing positions of the objects in the game.

The above screenshot is of the game play. On the top of the game there is a bar where the number of arrows remaining and the score is displayed. In addition, the features activated is also shown.

Here, the score of the player is shown. If the score is greater than the high score, than the high score gets updated.



4. Project Modules

- **Change of game states to navigate through pages**

  We used separate void functions for different pages/displays of the game like the main menu page, high score page, the instructions page and the actual game itself. This was done by assigning a gamestate value for each page. The gamestate value at times changes manually if a button on the screen is pressed or automatically if the game gets over.

- **Move.h**

  This is a custom header file. In the functions up, down, right, left, the SDL_Rect values are passed by reference. The value of box speed is passed by value. The functions are void; in each, the x coordinate and the y coordinate of the boxes are changed accordingly.

- **File.h**

  This is also another custom header file that makes use of the header library <fstream>. Firstly, here we have two structs – Mousexy and Buttonxy. Buttonxy struct has two sets integer variables of x and y representing the coordinates of the top left and bottom right corner of a rectangle (buttons in the game). The Mousexy struct has one set of integer variable x and y representing the position of the mouse click. The integer function button pressed checks whether the mouse click position falls into the area of the rectangle button. The void function update checks whether the current score is greater than the highscore from the text file-highscore.txt. If it does, it creates a new text file-temp.txt and stores the current score. The file highscore.txt is deleted and the temp.txt file is renamed to highscore.txt. The void function Reset() does almost the same thing as Update(), except here there is no condition involved and the high score is updated with the value zero. Lastly, the integer function get_highscore() just reads the high score value from highscore.txt and returns the number.

  Loadbuttons

  This is a void function that makes use of the text file-coordinates.txt. In that file all top-left corner and the bottom-left corner coordinates is stored. Using this function, we assigned these coordinates to the struct button's coordinates.

- **SDL_Get_Ticks() for timed feature activation**

  This SDL feature is used for activating and deactivating some of the game features within a specified time intervals was a part of SDL.h. Initially the integer variable start is initialized to 0. When a certain feature is activated the start variable is updated with the current time using SDL_Get_Ticks().The SDL_Get_Ticks() is then used to check the current time; if it exceeds the required time which is equivalent to the difference between SDL_Get_Ticks() and the start time then the game feature is deactivated.

- **SDL_ttf for printing strings on the game**

  We used SDL_ttf.h for printing characters and strings like number of shots remaining, score, highscore etc. We did this by first initializing a color – black. Then we converted integers to strings using function to_string() and in turn converted them to constant char using c_str(), which are basic stdio.h functions. We used external true type font files for changing font styles.

- **Randomizing the starting position of boxes**

  This module is used to randomized starting position of boxes using the function rand(). This function falls under the header library time.h. It helped us to increase the randomness nature from which the boxes would appear and also for the boxes to be spread out enough.

- **Key Board presses/Mouse clicks**

  Both the button pressed and mouse clicks were part of SDL.h. The key presses were detected using SDL_KEYDOWN as the event type and the specific keys were checked using event.key.keysym.scancode. The keyboard was used in the game for navigating the archer and for shooting the arrows. Moreover, the mouse clicks were detected by changing the event type to SDL_MOUSEBUTTONDOWN and the specific buttons were matched using e.button.x and e.button.y. The mouse clicks were used to press buttons in the game and navigate through the pages of the game.

- **Structs – box, featues**

  In the struct box, we initialized five variables. SDL_Rect box array is used for the five types of boxes of the game. The next integer variable – point signifies the point each box yields. The int variable width and height is for the box's dimensions. In addition, an array of flag is used to mark if the box has been shot.

  The struct feature is almost the same as the struct box. The 32 bit unary integer start is used to mark the starting time when a feature is activated. The const char* message is used for the message that would be printed on the top bar when a specific feature is activated. The variable flag is used to mark which type of box is activated/shot and the array flag_n is used to mark which box of which feature is shot. Lastly, the integer variable flag_mess is used to mark which message is to be displayed on the top bar. There are nine different features.

- **Using functions to adjust the string/numbers**

  The function redefine, redefine2, redefine3 uses the header library <string> to provide some adjustments to the screen like adding leading zeros to make all numbers three digits. It also involves adding spaces to the string so that the characters are aligned properly.

- **Render background images on screen**

  We used SDL_image.h to render images. Some of these images involve the background images. Each page has a separate background image that fits the entire window.

- **Render foreground images on screen**

  We used SDL_image.h to render foreground images like the archer, boxes, arrows etc. Unlike background images, these images had a specific dimension and a position on which the image would appear. Also, some of the images are associated with a condition of flag's state – that is the image only appears when the flag is true.

5. Team Member Responsibilities

Modules implemented by Md. Sadmin Tahmid Khan

- File.h
- Move.h
- Change of gamestates to navigate through pages
- Loadbuttons
- SDL_Get_Ticks for timed activation of features
- Struct- box, feature
- Rendering foreground images on the screen

Modules implemented by Ishwor Dhungana

- SDL_ttf for printing strings on the game
- Randomizing starting position of boxes
- Use of Keyboard presses and Mouse clicks
- Using function to adjust number/string
- Rendering background images

6. Platform, Library & Tools

We used the language C++ and the platform we used to make the game is SDL

7. Limitations

Our initial idea was to use balloons instead of boxes. The balloons were supposed to be shown to be burst and later fall below. However, we could not replicate the idea into the game because we did not find suitable images for the animation of a balloon burst. However, using boxes was not a bad replacement as it allowed us to apply new features to the game. In addition, we also planned to add a feature in which the number of arrows remaining could be altered. But again, we could not do it because of complicacy of the code. Instead, we added other features. There is one problem in our game and that is sometimes when a box is shot, the box does not disappear. We could not find the solution to this problem.

8. Conclusions

Working on this project allowed us to explore and learn a variety of ways in which codes can be made more efficient and most importantly we learnt several tools that can be used in SDL. Few of the things we learnt was to use file, custom header files and setting the time for which a code must be executed. Initially, when the idea of the game was pitched, it was not much interesting. But, later on, using new ideas and adding new feature we were able make the game more interesting. We changed the outlook of each window to make the game more appealing. Moreover, we added buttons to allow the user to change the window of the game, as well as, adding a window for High score and Instructions. Our main difficulty

was to find suitable images for the game. At times we needed transparent images that were difficult to collect. We even faced difficulty learning new was to innovate our game through, combining both of our codes and using SDL_ttf. But all in all, we are satisfied with the outcome of our work.

9. Future plan

Our Future plan of this project is to convert this SDL game into a mobile game. We have already looked up into what ways an SDL games can be converted into a mobile game, including the software we might be needing. We will convert the key presses and mouse clicks into finger taps and finger swipes for the game. Furthermore, we will change the dimensions the game and the displays as well. When necessary improvements are made we will convert the game into an Android mobile app game and release the game in Play store.

Repositories

GitHub Repository:

https://github.com/Sadmin23/Bow-And-Arrow-Game/tree/master

Youtube Video:

https://youtu.be/J_MNUuDZsUo

References

https://lazyfoo.net/tutorials/SDL/

https://www.gamedev.net/

https://www.sdltutorials.com/sdl-tutorial-basics