

Duck Hunt

Project 2

Candidate 162217 | Multimedia Design and Applications | 11th May 2017

Masters Advanced Computer Science

# 1: Introduction

This is a Duck Shooting game that has been designed in 2d, but given a sense of depth by allowing the ducks to fly away from the hunter, and get smaller as they fly into the distance. The scene gets darker as the day draws on and rain falls from the sky. When the hunter can no longer see to shoot the ducks, the level comes to a close and a new Day starts. This report will demonstrate an extensive use of programming to control the movement of ducks, and show image manipulation, particle rain, sprite animation, use of sound and control over several canvas levels. HTML5 and Canvas [3] with JavaScript will be used to write the game.

At the start of the game, an information message will inform the player that the game will only run in the Firefox Browser. For best results, use a screen resolution large enough to display a 1280 x 899 image.

# 2: Design and Development

Original Idea

The initial design was to create a carnival duck shooting game, with ducks revolving on disks. Ducks would take off and fly in and out of particle clouds of varying size and complexity when they heard a shot. Ducks would get smaller when they fly back into the picture giving a feeling of depth. When they sense the danger has passed they will land and re-attach themselves to a random disc with a vacant spot. When all bar one of a group of ducks had died, the last duck living would attack the hunter, who would lose confidence if the attack were successful. The hunter could regain some of his confidence by shooting the attacking duck. The ducks could also win the game if the hunter runs out of confidence after several attacks and runs away.

Change of Emphasis

The original idea had too many features, it would have taken too long to complete and it would have received a mark well past that required for distinction. Having re-read the project 2 description, I decided to split the work between features / functionality and an extensive use of programming/scripting.

The design started with an idea… Wouldn’t it be great if the ducks occupied and moved around in a 3D space, but were actually shown on a 2D image. From this idea came the idea of depth.

The game starts with a title page that welcomes you to the game. Just follow the instructions to start the game.



Figure 2.1 Initial Welcome Screen

The Background Image:

The need for a background image with well-defined areas going back into the distance influenced the choice of background image used. There is a large pond at the bottom of the image with a low border area of reeds and low bushes, with trees in the distance and clouds in the background [6]. This gave a real feeling of depth that made it easy for me to split the scene up into well-defined areas where ducks could fly back into the distance.

As the game progresses the background image gets darker to reflect the passage of time, this was achieved by extracting the Image data and manipulating the pixel RGB colours.

Rain Particles:

Corel Paintshop Pro was used to create three different sizes / shades of raindrop. The background of each raindrop was made transparent and saved as a PNG file. 230 raindrops with random sizes and starting positions on the x-axis were then created and stored in an array ready for display. The raindrops were then animated [1].

The Environment:

The background has been split up into four levels on the face of the background image and four depths that go back into the picture. As ducks move around, they have the option of going backwards and forwards through depths and up and down through levels. The current level and current depth of each duck are stored and updated as the ducks moves to ensure they do not exceed a boundary limit. As ducks go between the depths in the picture, they also move up between the levels. The larger ducks at a depth of ‘0’ can roam freely between all levels of the canvas, the smallest ducks at a depth of ‘3’ can only fly in the top level of the image. Figure 2.2 highlights the zones available for the ducks to fly into.

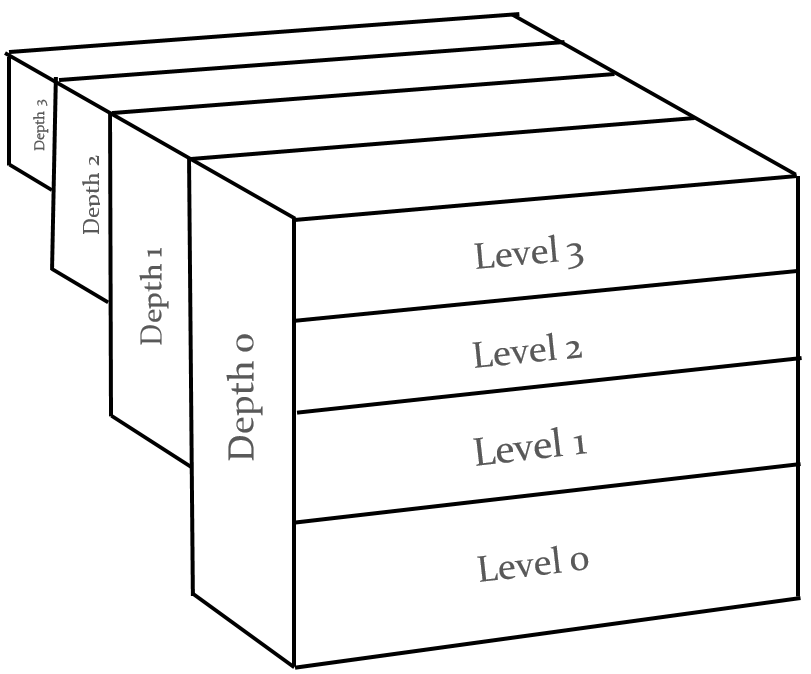


Figure 2.2 Game Environment

The Actors:

The actors are split up into three different types of duck (Green, Red and Blue), each type of duck has 17 different sprites associated with it. The sprites were created using “Coral Paintshop Pro X8” on a 32 x 32 grid, the background was then made transparent before being exported into a PNG format. The sprites are split up into groups of three of four frames to show a particular movement when animated [4]. The ducks are created using Adobe Illustrator. The ducks flying left are clones of the ducks flying right. The images were turned into High fidelity photo images and an image trace was done to separate the duck from the background, the background was then made transparent. [5]



[ Fly Right ][Fly Up and Right][ Fly Left ][Fly Up and Left][##][ Dying Duck ] . [Down and Right] [Down and Left]

Figure 2.3 Duck Sprites and their Movement

Each duck has a different sound file associated with it [8], every time a duck changes direction it will call out. The volume changed depending on the current depth the duck is in. This allows ducks that are closer to the front to be louder than those ducks in the background. When a duck is killed, the call of the duck is stopped, so the scene becomes quitter as the ducks are killed.

Movement:

Ducks have a wide variety of movements available to them. An array of possible next moves is created based on the direction of the previous move and the possible moves that can be made from this direction of flight. If the duck is found to be at the minimum or maximum depth, a choice is added to the array to move away. A random movement is then selected from the list of options. A bias can be placed on one or more of the movement directions to help control the movement of ducks around the screen.

The example below shows the list of random ‘next moves’ that could be selected if the previous move made was ‘Left’.

Options: (Left, Left, Left-to-Right, Left-and-Up, Left-and-Down, Left-and-Down)

Add ‘Left-and-Forward’ if Duck not at minimum depth

Add ‘Left-and-Back’ if Duck not at maximum depth

Add ‘Left-and-Back’ if Duck is at the minimum depth

Add ‘Left-and-Forward’ if Duck is at the maximum depth

Figure 2.4 Random Movement Selection

Please note the duplication of ‘Left’ and ‘Left-and-Down’ options to give a bias to move in these directions.

For each direction of movement, a repeat option can be applied. This could be a random number of repeats within a range or a set number.

There is a different list of ‘next move’ options available if a duck is trying to avoid another duck, this is because the emphasis has changed to ‘moving away’ from the duck rather than ‘allowable moves from a previous direction’.

Movement types can be linked to other movement types, so that sequences of movements can be created like the ‘Right-to-Left’ option that will fly up to the left, and then fly up to the right before flying right. There is no limit to the number of moves that can be linked together, but this sequence will end during Duck or Edge avoidance.

Each movement type also has a set trajectory and distance associated with it, with different x and y movement amounts in pixels for each depth of duck, so that large ducks can move at different angles and distances to smaller ducks.

I decided on this algorithm for the movement of ducks because it lends itself to change. The movement of ducks can be analyzed, and changes quickly applied to modify behavior and make the movement seem more realistic, or chaotic depending on what you are looking for. Full control is given to movement by changing a number of variables

Collision Detection:

Ducks are only allowed to go between depths into the canvas by moving backwards or forwards. If they are moving up or down within a zone a check has to be made to make sure the edge of their current zone is not about to be reached. The table below checks the maximum y coordinate allowed for a duck in a certain area and if their subsequent move would take them past the maximum the move is replaced with one that takes them away for the Edge.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Zone Edge Detection Table | | | |
|  | Depth 0 | Depth 1 | Depth 2 | Depth 3 |
| Depth End | 899 | 676 | 451 | 226 |
| Depth Edge | 30 | 25 | 17 | 8 |

Figure 2.5 Zone Extremity Detection

Ducks also do their best to avoid other ducks at the same depth to themselves. If ducks are on different levels the ducks at the highest depth will always appear under ducks at a lower depth that are closer to front, this makes the scene more realistic.

Shooting the Ducks:

A red rifle crosshair (Figure 2.6) now replaces the mouse pointer [2]. You shoot the rifle by clicking the mouse button and the sound of a rifle can be heard [7]. You score points by shooting a duck while the duck is under the crosshair.



Figure 2.6 Crosshair Rifle Site

The amount of points you score depends on where you hit the duck and the size of the duck being hit. Each duck is split up into two shooting zones (Head and Body). The coordinated for both shooting zones are stored in different arrays that are checked every time a shot has been made to see if the coordinates for the gunshot are within the coordinated of either a ducks head or body. To separate the coordinates for the head and the body the x,y coordinated for the top left and bottom right for each head and body are stored in arrays.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Large Duck Depth of Duck Small Duck | | | |
|  | Depth 0 | Depth 1 | Depth 2 | Depth 3 |
| Head Shot | 150 | 170 | 190 | 240 |
| Body Shot | 100 | 120 | 140 | 160 |

Figure 2.7 Points awarded per hit.

# 3: Conclusion

Did the project achieve its aim?

If the aim was to complete the original specification, then no it did not achieve its aim. However, the original ideas would have taken too long to incorporate, and they exceeded the level required for a distinction. The change of emphasis was the right decision with a mix of features/functionality and extensive use of programming/scripting. In my opinion, the game achieved its reevaluated aim.

Was the project successful?

Most of the hard work went into the movement of ducks and giving a perception of 3D on a 2D image. By splitting, the background into areas of depth that ducks could fly back into while getting smaller, and fly forwards out of while growing larger, a perception of 3D space seems real. An extensive selection of movements and creation of movement sequences also help add to this perception. The design of the program affords ease of fine-tuning of movements. Given time and some tweaking an even more realistic visual experience could be created. Adding individual sounds to each duck where the volume goes up and down while flying back into the image, of flying forwards also helps the perception. Given the perception of 3D, extensive programming for movement, animation, image manipulation, and particle rain generator, I believe the project was successful.

What Future Enhancements would be of benefit?

Given more time, I would love to have incorporated particle clouds into the design. Shown the reflection of the clouds in the water. Created ripples in the pond when a duck hit the water. Created other objects like balloons coming into view that could be shot, and either explode killing surrounding ducks, change the rifle to a machine gun, or launch a missile that could be controlled via the keyboard. Adding a complete eco system with Thunder, and lightning would also be great. In addition, allow ducks to attack the hunter and possibly win the game. There should also be a high score table. Allow score to be uploaded to Facebook, Twitter etc. The hunter should be able to sell the ducks and buy goods / upgrades from a store. (Trained hawk controlled via keyboard, flamethrower, bird table and seed).

1723 lines of JavaScript source code were written to produce this game

# 4: References

1: CSS-TRICKS Keyframe Animation Syntax. Available at: https://css-tricks.com/snippets/css/keyframe-animation-syntax/

2: RealWorld Graphics - web and application graphic resources. Available at: http://www.rw-designer.com/cursor-detail/42287

3: Steve Fulton 2013. HTML5 Canvas. Available at: http://chimera.labs.oreilly.com/books/1234000001654/index.html

4: WEBAPPERS How to Make Sprite Animations with HTML5 Canvas. Available at: <http://www.webappers.com/2013/02/08/how-to-make-sprite-animations-with-html5-canvas/>

5: Corel Support knowledge base. Available at https://support.corel.com/hc/en-us/articles/216435177-How-to-create-an-image-with-a-transparent-background

6: Wallpaper Abyss – Alpha Coders Pond Wallpapers. Available at: <https://wall.alphacoders.com/by_sub_category.php?id=174663>

7: FREE SOUND EFFECTS Gun Sounds. Available at: <http://soundbible.com/tags-gun.html>

8: Maestro Game Calls. Available at: http://www.gsmoutdoors.com/maestro/digital-sounds/duck-mallard/