SNAKE 3d

interactive graphics' project

A.A. 2016-2017
Luca Borzacchiello 1602625

Table of Contents

1. Introduction

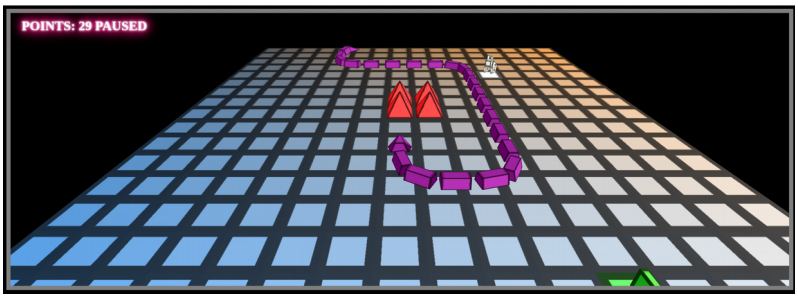


Illustration 1: Game screenshot

The software produced for this project is a 3D version of the famous game *snake*. In *snake* the player controls a snake which has to eat as many prey as possible. When the snake eats a prey, its body becomes longer and the player gains an amount of point which depends on the type of the prey. The player looses if the snake hits himself, hits an obstacle or steps over any of the edges of the environment.

In this particular version there are two types of preys: *The Rabbit* and *The Parallelepiped* whose technical aspects will be described in the following section of this report. The Parallelepiped just stands in the same location until the snake eats it. The player gains one point when the snake eats it. The Rabbit, on the other hand, moves randomly in the environment. If the snake eats the Rabbit, the player gains five points. Both prey are generated again in a random location of the environment as soon as the snake eats them.

This version of the game has one level and the speed of the snake is fixed but the code can be easily modified in order to change both the environment and the speed of the snake.

The player can interact with the game using the arrow keys (to move the snake) and with the key P in order to pause the game.

2. Technical aspects

The technologies which I used to develop the project are only HTML5, Javascript and WebGL. The only external library which I used is the one by Ed. Angel from the textbook [Angel's book].

As support for 3D modeling, I've used the software Geogebra [Geogebra].

2.1. The Snake

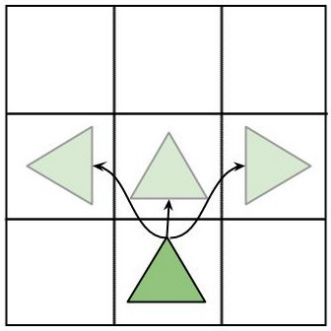
Illustration 2: Snake

The snake is the main element of the game and is controlled by the player. It is formed by three parts: the head, the body and the tail. When it eats a prey, a new body part is generated.

The behavior of the snake is simple: the player controls the head and the first body part moves according to its actual position and according to the head movement; all the other parts simply copies the last movement of the section which is ahead of them. Technically the snake is implemented with a linked list which is recursively scanned at each step of the animation.

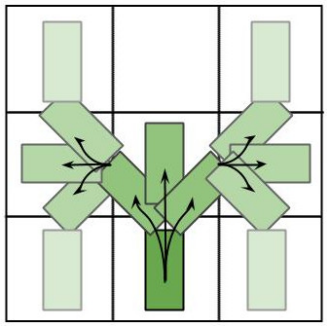
The head of the snake can move only in three ways [drawing 1]:

- *Forward movement*: triggered by default if no arrow key is pressed by the player.
- *Left rotation*: triggered by the left arrow key, the head rotates counter-clockwise by 90 degrees and translate by one square forward and one square left.
- *Right rotation*: as before but in the other direction.



Drawing 1:Head movements

The body and tail movements are a bit more complex, depending on their previous position and depending on the head movement, they can rotate by 45 or 90 degrees



Drawing 2:Body/tail movements

and they can translate forward by 1 square or less/more, depending on the situation [drawing 2].

2.2. The Rabbit

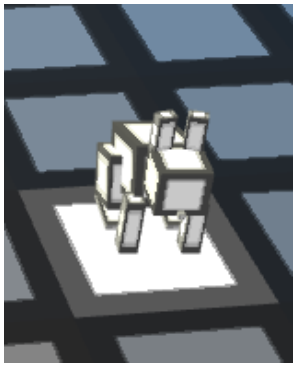


Illustration 3: Rabbit

The Rabbit is one of the prey and is the object with the most complex hierarchical structure [drawing 3] in this project. It is formed by 11 parallelepipeds with one degree of freedom each.

The Rabbit can jump forward or rotate by 90 degrees left/right.

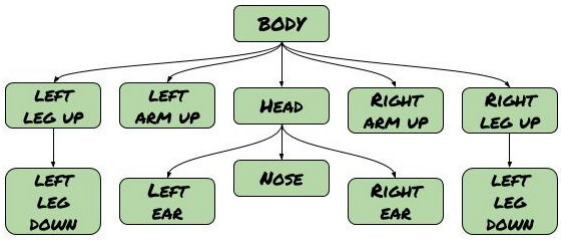
It moves randomly in the environment according to Algorithm 1.

```
if last movement was rotation and it is possible to move forward then
      move forward
else
      with 50% of probability move forward
      with 25% of probability rotate right
      with 25% of probability rotate left
```

Algorithm 1: Rabbit movement

The rotation movement is very simple, the rabbit just rotate around itself by 90/-90 degrees.

The animation of the jump is quite more complex, it is broken down by three phases [illustration 4]:



Drawing 3: Rabbit hierarchical model

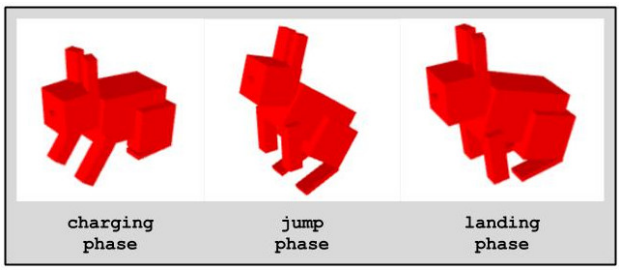


Illustration 4: Rabbit jump

- *Charging phase*: in this phase the rabbit pull forward its arms and flexes its lower legs.
- *Jump phase*: in this phase the rabbit jumps, relaxing its arms and legs.
- *Landing phase*: in this phase the rabbit lands softly in a new square.

2.3. The Parallelepiped and the Obstacles

Illustration 5: The Parallelepiped

The *Parallelepiped* is the other prey which the snake can eat. It just stands in one square of the environment, rotating around itself waiting to be eaten by the snake.

The 3d model is simply a parallelepiped. As soon as the parallelepiped is eaten, a new one is generated randomly in any free square of the environment.

The *Obstacles* are just pyramids which stand in a square of the environment preventing the snake to pass through them. They cannot be destroyed and remains in the environment for the whole match.

Illustration 6: Obstacle

2.4. The Environment

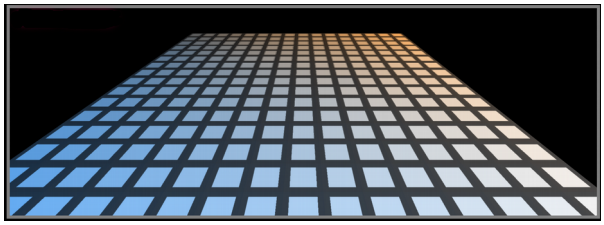


Illustration 7: Environment

The environment is the grid which contains all the object of the game. The snake and the rabbit move in the environment and at the end of each animation, all the object stand in one and only one square.

All the squares are rendered separately at the begin of each animation and the shading effect is achieved through textures and colors.

Programmatically, the environment is modeled through a matrix, each cell of the matrix contains an object which represent a square of the environment. The data that the object contains include: the color of the square, the type of object which is on the square (if any), the model-view matrix to be applied to the square at render time, etc.

2.5. Camera, light and textures

The *camera* is placed above the snake and remain fixed in the coordinate system of the snake head, so with respect of the environment it rotates and moves according to the snake.

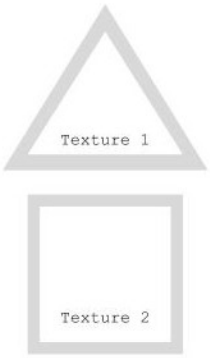


Illustration 8: Textures

There is only one *light* in the environment, a directional light which hits the squares with an angle of 45 degrees. The light is computed using Gouraud shading. Other light effect (such as the shading of the squares) are achieved through texturing and coloring.

The only two *textures* used in this project are the ones in Illustration 8. The textures are applied to every faces of every object and are multiplied in the shader with a color. Each object is characterized by a specific color.

The textures are computed programmatically.

2.6. Game loop

The game loop is divided in two different parts. During the first part, the current animations are simply displayed for each object; during the second part it is checked whether or not a key was pressed, all animations of the new first part are decided for each object, it is checked if the snake has captured a prey etc.

The game loop is repeated until the player looses.

Bibliography

Angel's book: Edward Angel, Dave Shreiner, Interactive Computer Graphics with WebGL,
Geogebra: Markus Hohenwarter, others, www.geogebra.org, 2001-2017,