DIAMONDS VS. HEXAGONS

Computer Graphics Assignment

Șerban Andrei-Teodor

334CC

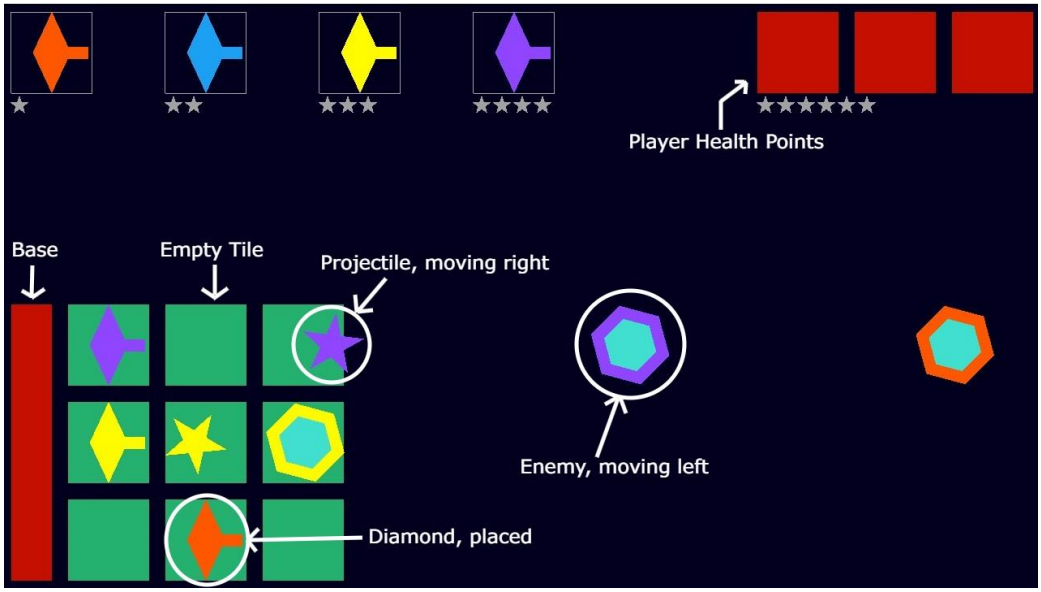
TABLE OF CONTENTS

0. General

This document describes the content and implementation of “Diamonds vs. Hexagons”, a 2D game written for a Computer Graphics assignment [1]. The game is inspired by Plants vs. Zombies, a popular tower defense title from 2009. Both the aforementioned video games have similar mechanics.

You have a base in the left-hand part of the screen that you need to defend from attackers. In the case of this project, the attackers are hexagons. The enemies move in straight lines from the right of the screen towards your base and if they touch it, you lose a health point. In order to keep the diamonds away, the player has a grid of tiles in front of the base. On the tiles you can place diamonds, which shoot projectiles (i. e., stars) on the lines that the hexagons are coming from. The game ends whenever the user’s base has been hit by a hexagon three times.

The game is written in C++ and uses OpenGL [2] and the gfx-framework [3] for the purposes of understanding the graphical pipeline via a lower level application.



1. Scene

This section expresses how the scene is rendered upon launching the game.

The game elements that are rendered before any of the dynamic behavior begins are the following:

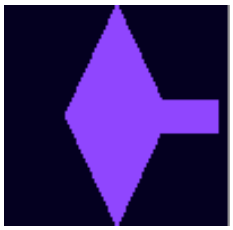
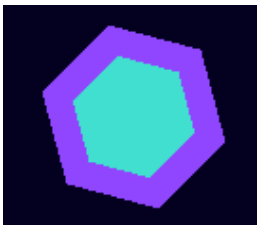
- A 3x3 grid of green tiles, representing where the player can place diamonds
- A red rectangle, representing the player’s base
- Four bordered diamonds, representing the types of diamonds that the player can choose from
- Three red squares, representing the player’s health points

All of these static elements are rendered from basic pre-loaded square meshes that only needed to be imported into the project.

2. Elements

This section describes how the more complex visual elements are constructed for the game.

For these graphical items, except the diamonds, the meshes are loaded manually with vertex and index vectors, using tools provided by the gfx-framework. More precisely, each point of an object's geometry is defined as a point in 2D space with coordinates and an order in the topology. The topology uses GL_TRIANGLE as the base graphical primitive, so the indices are defined in groups of three.


2.1. Diamonds

The diamond is the only dynamic object that is not rendered by manually defining each point. It's fully rendered using two base square meshes that are scaled and rotated to create the final shape.

The first mesh is rotated 45 degrees and then scaled unevenly to create the diamond shaped body. Both of these 2D transformations are made relative to the center of the original square. After that, the second mesh is scaled down on the y-axis for the width of the cannon. Additionally, the mesh is halved on the x-axis and translated to the right.

2.2. Hexagons

For the hexagon's vertices, the first point defined is the center. The hexagon consists of six identical triangles, which are all concurrent at the shape's center. From the center, a radius is chosen and six points on that trigonometric circle are set afterward. Since the points are equidistant, they are $\frac{2\pi}{6}$ radians apart. As a result, the (x, y) coordinates of each of these points are defined iteratively, using cosine and sine respectively for all the different angles.

During the game, each hexagon mesh is rendered twice to display a smaller hexagon inside the colored one. The movement of each enemy object is a simple right to left translation, without any rotation.

2.3. Stars

The most complex shape in the game, the five-pointed star is the projectile shot by the diamonds to destroy the hexagons. It is defined using the vertices of two pentagons of distinct radii. The ratio of these radii is $1 + \varphi$ [4]. The construction of the mesh is similar to the hexagon, except the distances between the points are $\frac{2\pi}{5}$, since the shape is built from pentagons. In contrast to the hexagon, the topology only needs to define three triangles. These triangles are defined using two points from the large pentagon (forming a line) and the point opposite to that line on the small pentagon. Choosing any three triangles with this method fills out the star.

When rendered, the stars are translated from left to right and a clockwise rotation relative to their center is applied.

3. Behavior

This section outlines the finer details of how the game evolves over time and how the user interacts with it.

The game's controls require only the mouse in order to be played. The diamonds in the interface are interacted with via drag and drop mechanics. The player can use LMB [Left Mouse Button] to drag the diamond from the select menu at the top of the screen and place it on an empty tile. The shooter will not throw projectiles if its line is not being attacked. Whenever the line is attacked, the diamond shoots projectiles at a three second interval. Every enemy requires three hits in order to be destroyed, at which point it scales down in size until it disappears. If an enemy collides with a diamond, the diamond scales down similarily to the hexagon's death. If three enemies cross the tiles and arrive at the base, the game ends, rendering a game over screen.

The enemies and shooters are displayed with 4 distinct colors:

- Orange
- Blue
- Yellow
- Purple

The lines and colors of the enemies are randomly generated. Additionally, they appear every five seconds.

One element of a certain color can only interact with other elements of the same color. For instance, a yellow diamond will shoot yellow stars only when a yellow hexagon is attacking that line. The projectiles do not interact with hexagons if they are not of the same color.

3.1. Collisions

Collisions occur either when a projectile hits a hexagon, or when a hexagon arrives at a tile where a diamond is placed. Additionally, player LMB clicks can be considered collisions with either the UI diamonds or the tiles.

With the diamond-hexagon and star-hexagon collisions, the implementation is the same. Diamonds, hexagons and stars are approximated with encompassing circles. If the distance between the centers of two of these elements is smaller than the sum of their radii, then the collision occurs.

For the mouse click collisions, another principle is used: axes-aligned bounding. This means that the exact coordinates of a click are checked to be within a certain square, in which case the collision is detected.

3.2. Deleting a Diamond

If a player uses RMB [Right Mouse Button] to click on a tile with a diamond placed on it, the diamond gets deleted. The result of the deletion is the same as if the diamond were destroyed by a hexagon: a scale-down animation until it derenders completely.

3.3. Pick-ups

Another aspect of the game consists of the smaller white stars, which render in a random position every seven seconds. These are pickups for the player, acting as a currency with which they can buy more diamonds to place on the tiles. All of the details about the game's economy are displayed in the UI at the top of the screen. Under each diamond there is the number of stars required to buy it. Under the user's health points the number of stars currently held is displayed.

4. References

[1] https://ocw.cs.pub.ro/courses/egc/teme/2023/01

[2] https://www.opengl.org/

[3] https://github.com/UPB-Graphics/gfx-framework

[4] https://math.stackexchange.com/questions/2136781/ratio-of-outer-circle-to-inner-circle-diameter-in-pentagon-star