Powdergame

Ben Lewis-Jones (N0928518) – SOFT10101 – Project Report

Table of  
Contents

Emphasis Heading 1 1

Heading 2 1

Heading 2 2

## Specification

My idea a simple falling sand game, where the player can draw or place 2D pixelated sand or water. The concept is for the different materials to interact with each other and its environment. There is no objective to the game other than your creativity. The game will also include the ability to save previous snapshots of what you have made, so you can play with it again at a later date.

This table will show what aims I had set for the project:

|  |  |
| --- | --- |
| Functional requirment number | functional requirement description |
| FR 1 | User should be able to place/draw materials into the window. |
| FR 2 | Materials should be able to interact with each other (not fall though each other). |
| FR 3 | User should be able to save and load snapshots of the game, this means where all the martials are and their current state. |

## design and Implementation

### Classes

Text

Description automatically generated

The Frame class contains all the methods and attributes needed to load a window onto the screen.

A picture containing text

Description automatically generated

This class is closely tied to the Window class, this class will poll though all the events, and through a series of getters will message them results to other classes.

Graphical user interface, text, application

Description automatically generated

This class holds all the information on a particle, this is done using structs, I chose structs because it will make my code easier to edit in the future, for example add more objectStatus variables.

Text

Description automatically generated

This class will create a grid and based upon the values of the attributes of each Particle object. This is achieved with the use of an object vector, I chose an object vector over an object array for its mailability using the push and pop functions.

Text

Description automatically generated

This class creates buttons on the screen, and depending on the type set at initialization will perform different actions, i.e. save the object vector to a text file, load the object vector from a text file, or change the type selected to paint different particles to the display.

Diagram

Description automatically generated

This class inheritance diagram shows the two types of particles, sand, and wall, they both share common code, this is used by both and is called the swap attribute.

They also have their own unique code, which is the action attribute, this determines what the particle is going to do on the next frame.

Diagram, schematic

Description automatically generated

### Challenges

I faced some issues with linking classes together, sometimes it seems most logical to message between objects, however this poses a large problem. You cannot call attributes from both classes, since you would need both classes to already be defined, this can cause a compiler error.

To resolve this, I had to rethink how to create my classes to avoid using back pointers and over complicating my code.

Another challenge I had to overcome was the ability to save objects to a .txt file, I resolved this by saving the individual attributes to the file and then reconstructing the objects when loaded.

The final main issue was to get the window to display on the desktop, the error was showing a window however it was void of contents and was completely transparent, only showing the window borders. The solution was that MacOS only displays a window if it has a quit function, thus I implemented a quit function in the event loop.

## testing

#### Test Plan

|  |  |  |
| --- | --- | --- |
| Action | Test Method | Success Criteria |
| Add sand | Press on sand button (yellow) and then press on the particle display. | Sand appears on the display and interacts with the environment correctly. |
| Add wall | Press on wall button (grey/brown) and then press on the particle display. | Wall appears on the display and interacts with the environment correctly. |
| Remove particle | Press on remove particle button (black) and then press on the particle display, where there are particles. | Particles should be removed from the particle display. |
| Save current particle display | Press on the save button (red). | Current particle display should be saved to the file.txt. |
| Load current particle display | Press on the load button (blue). | Particles from the save should reappear in the same format. |
| Exit | Press on the exit button (top left X). | Program should close. |
| Particles interact correctly | Using the wall and sand particles next to one another. | They should flow through each other. |

#### Results

|  |  |  |
| --- | --- | --- |
| Action | Results | Final notes |
| Add sand |  | Sand was added correctly. |
| Add wall |  | Wall was added correctly. |
| Remove particle |  | I removed particles from the test above, thus was successful. |
| Save current particle display | Text file is too large to show in this document | The correct data was added to the text file. |
| Load particle display | Before:  (this was now saved)    (Then sand was added)    After:  (pressed load) | This test was successful. |
| Exit | The program exited | Test was Successful. |
| Particles interact correctly |  | The sand didn’t go through the wall; thus test was successful. |

## Evaluation

In future versions of the program, I would like to add more particle types, such as water; I would also like to implement a better way of rendering the particles, since you cannot see particles falling in this current version, this is caused by the way the screen is rendered.