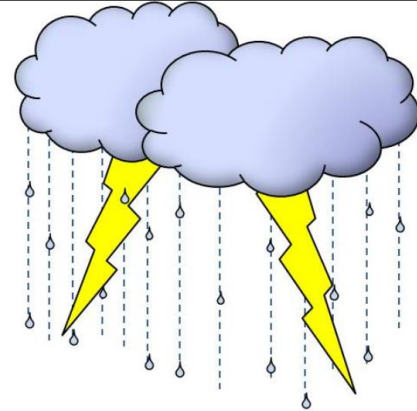


Sunny



Cloudy



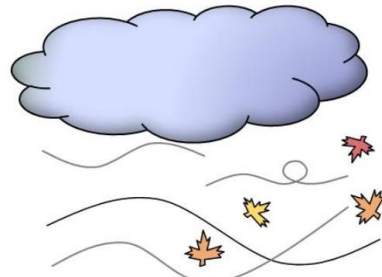
Stormy



Snowy



Rainy



Windy

Weather Engine 2D

Building Game Engine CS5850
Project Report
04.18.2018

Harshit Gupta

gupta.hars@husky.neu.edu

Northeastern University
360 Huntington Avenue
Boston, MA 02115

Ishan Patel

patel.ish@husky.neu.edu

Northeastern University
360 Huntington Avenue
Boston, MA 02115

Tanmay Sinha

sinha.t@husky.neu.edu

Northeastern University
360 Huntington Avenue
Boston, MA 02115

Akash Parikh

parikh.ak@husky.neu.edu

Northeastern University
360 Huntington Avenue
Boston, MA 02115

Overview

THE GAME

Our goal was to create a Weather Engine for our platformer game, using which the user can easily add different weather simulation to the game. Weather simulation would also lead to certain changes on the game objects.

Project Link: <http://ishanpatel.in/weather-effects-engine/>

GOALS AND EXPECTATIONS

We decided to create different weather simulations such as Rain, Wind, Sandstorm, Snow and combined impact of wind and rain. We were a team of four members and we firmly believe that our game is above average in terms of innovation, quality and creativity. People who tried our game gave positive feedback.

THE PROCESS

We did everything as per the course requirement. We started with creating a Weather class which needs to be imported by the platformer class to simulate different weather effects. In weather class we have added functions to add wind, rain, snow and sandstorm. Apart from adding this function we have also allowed developer to change the direction of wind or increase the density of wind or rain or snow which in turn creates more number of rain drops on the screen. The user can also add different sound effects for each weather functionality.

RESULT

The weather functionality is fully functioning and could be easily checked by the user using a very user-friendly interface. Weather functions also impacts the game objects like adding wind feature creates default user movement in the direction of the wind. Adding sandstorm creates default user movement in forward direction to get rid of sand.

Setup

To View Source Code:

1. Go to Platformer folder
2. Go to include folder to get weather.h file

3. Go to src folder to get weather.cpp file

To View Demo:

1. Go to Platformer folder
2. Double click on Platformer.exe
3. Select the Weather Option you want to test

Failures and Setbacks

1. We tried to implement Lightning system but the Lightning effects were interfering with the map tiles. Hence we hope to put it in future update.
2. We faced problems while detecting collision between the object and rain particles. So we allowed the user to provide the dimensions of the weather effect area so that he can limit the weather affects to a particular area removing the need to collision detection.
3. The Snowfall behavior is still buggy when we try to replay the game.
4. We also wanted to add buttons on user interface which could easily enable the developer to change wind direction or density.

Successes

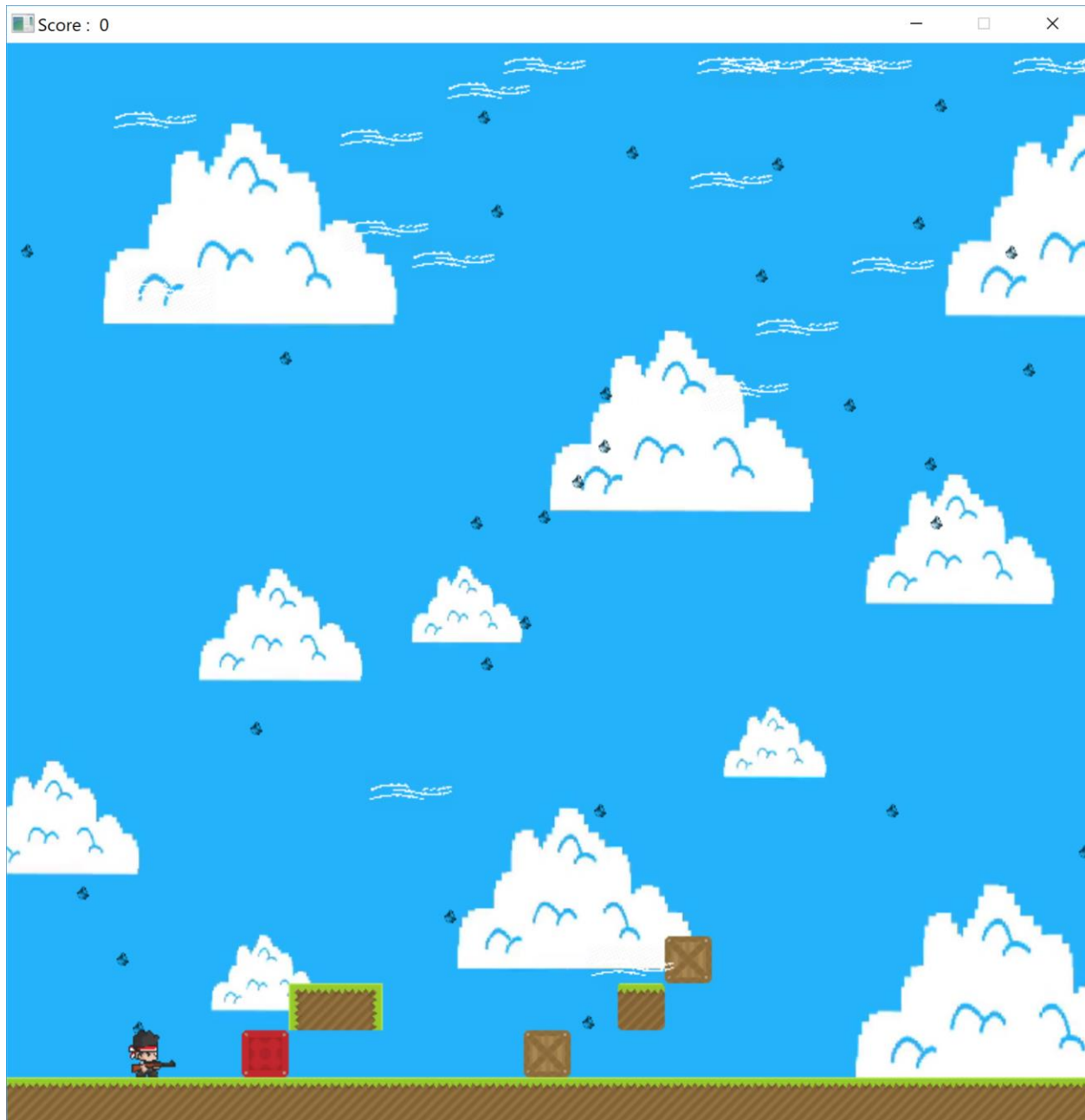
1. The Wind, Rain, Snow and Sandstorm behavior work as expected.
2. The effect of Wind on Rain, Snow and Sandstorm is evident and works as expected.
3. The effect of Wind on game objects like displacement in X direction is accurate and feels realistic.
4. Developer could also change the texture simply by changing the image from assets folder as we have implemented a data driven weather engine.
5. Randomizing the movement of sand, rain and snow particles give it a realistic effect

Lessons Learned

- Memory Management plays a big part during engine development. A small tweak in printing messages and debug can save a lot of memory.
- Always try to maximize the reusability of imported textures. During dynamic input of resources, improper management of resources may break a game.

- Testing with game development is difficult in that many features are dependent on each other that makes unit testing complex. For example, we need to have game dynamics implemented first before we can test collision or effect of a feature on other game objects.

Screenshot:



Attributes

Integers:

```
int windParticleHeight = 0, windParticleWidth = 0;  
int window_width, window_height;  
int origin_x, origin_y;  
int particleDensity;  
int windDensity;  
int rowCount, colCount;  
int TileWidth, TileHeight;  
int windDirection;  
int particleWidth, particleHeight;
```

Floating Point:

```
float windSpeed;  
float particleSpeed;  
float rainDirection;
```

Boolean:


```
bool windEnabled;  
bool rainEnabled;  
bool snowEnabled;  
bool sandEnabled;
```

SDL:

```
SDL_Rect* weather_particles;  
SDL_Rect *wind_rect;  
SDL_Texture *weather_texture;  
SDL_Texture *wind_texture;
```

Functions:

- **Weather(int window_origin_x, int window_origin_y, int windowHeight, int windowWidth);** Constructor
- **void setWindSpeed(float speed);**
set wind speed given by the user
- **void setWindTextureDimensions(int height, int width);**
set the size of wind texture
- **void setRainSpeed(float speed);**

- 
- set particle y-velocity given by the user
 - **void setSnowSpeed(float speed);**
set particle y-velocity given by the user
 - **void setRainDirection(float direction);**
set the direction of rain in x-axis
 - **void setParticleDimensions(int height, int width);**
set the size of snow or rain particles
 - **void rainFrequency(float frequency);**
set the density of rain
 - **void addWind(SDL_Renderer *r, std::string filepath, float speed, int direction, int density);**
enable wind effect by setting windspeed and populating the wind SDL_rects array
 - **void addSandstorm(SDL_Renderer *r, std::string filepath, float VerticalSpeed, float HorizontalSpeed, int direction, int density);**
add sandstorm particles to scene
 - **void addRain(SDL_Renderer *r, std::string filepath, float fallSpeed, int density);**
initialize rain particle array in order to add rain
 - **void addSnow(SDL_Renderer *r, std::string filepath, float fallSpeed, int density);**
initialize snow particle array in order to add snow
 - **void updateWeather(SDL_Renderer *r);**
update weather at each frame
 - **float windDisplacementX(float mass, float surface_area);**
calculate change in an object's position in each frame due to the effect of rain based on the object's mass and surface area in x-axis
 - **float windDisplacementY(SDL_Rect object, float mass);**
calculate change in an object's position in each frame due to the effect of rain based on the object's mass and surface area in y-axis