Asteroid Game

COMPUTER GRAPHICS LAB PROJECT

Submitted by
Jai Chaudhry 2K18/SE/069
Ishaan Jain 2K18/SE/067

Software Engineering
DELHI TECHNOLOGICAL UNIVERSITY
BAWANA ROAD, DELHI-110042
November, 2020

Abstract:

This project report summarizes the project implemented in Computer Graphics Lab in 5th Sem DTU, under Pratima Sharma Ma'am.

The goal of this project was to design a 2-D graphical computer game using SFML. For this Asteroid Game was chosen. The user, who plays as a spaceship controller, has to destroy maximum asteroids to get a high score. The game is designed in Windows Environment, on CodeBlocks and written in SFML and C++.Various classes like Player, Bullet, Asteroid were implemented. Images along with SFML function were used to make the game and explosions were rendered on screen like a video using from scratch implementation of animations.

Table of Contents:

Introduction:

Asteroid Game is a very popular game and was one of the first major hits of the golden age of arcade games. It has been developed on various platforms using various libraries in the past few years. So, it was decided to implement this using C++ and SFML to implement Computer Graphics theoretical concept Animation.

This report presents a way of describing the processes involved in making of a 2D Asteroid Game (Graphical Computer Game) using SFML with C++.

The task in this project is to write a keyboard-driven, single player version of the asteroid game.

The main aim of this project is to learn to computer graphics concepts by programming in C++ with SFML.

Motivation:

Computer Graphics is a very important part of our lives as everything we use has digital interfaces which not only makes it interesting but also an important subject to study and this study cannot be done using theoretical means only so we decided to implement a game using Computer graphics concepts like Window to Viewport Transformation, Image Cropping, Animation etc. to better understand the topic.

Also, Games are not a waste of time, as they can improve the motor skills of people thus educating them (in a way). A game can develop patience and tolerance attributes in a person which are highly desired in society.

One cannot develop a big game without developing a small game because all need to take small-small steps to eventually take a big step in any domain.

There are many movies like “Resident Evil”, “Lara Croft”, “Prince of Persia” etc. which are based on video games because of their intense popularity.

Description of the work:

This section gives an overview of the work done in this project.

Player, Asteroid and Bullet classes were made. Each class has two major functions namely settings and update.
Self-made Animation class was used to change frames of objects with time.

Coming to SFML, Textures and Sprites were used to load and display the images like background image which is different for different screens.

Frame rate was set at 60 for smooth rendering of frames.

The game has 3 main screens namely Landing Screen, Pause Screen and GamePlay Screen which is shown in further sections in this report.

During Gameplay, the asteroids are randomly generated in the window if the user destroys any asteroid and the saturation limit is currently set to 20.

Gameplay was developed in 5 weeks and GUI (Graphical User Interface) in 2 weeks and the total number of lines of code in the project is around 800.

Related Work:

The game was included as part of the Atari Lynx title Super Asteroids & Missile Command,[12] and featured in the original Microsoft Arcade compilation in 1993, the latter with four other Atari video games: Missile Command, Tempest, Centipede, and Battlezone.[13]

In 1998 Syrox Developments developed an enhanced version of Asteroids.[14]

The Atari Flashback series of dedicated video game consoles have included both the 2600 and the arcade versions of Asteroids.[15][16]

Published by Crave Entertainment on December 14, 1999, Asteroids Hyper 64 is the Nintendo 64 port of Asteroids. The game's graphics were upgraded to 3D, with both the ship and asteroids receiving polygon models along static backgrounds, and it was supplemented with weapons and a multiplayer mode.[17]

The arcade and Atari 2600 versions of Asteroids, along with Asteroids Deluxe, were included in Atari Anthology for both Xbox and PlayStation 2.[18]

Released on November 28, 2007, the Xbox Live Arcade port of Asteroids has revamped HD graphics along with an added intense "throttle monkey" mode.[19]

Glu Mobile released a mobile phone port of the game with supplementary features as well as the original arcade version.[20]

Background Details:

Simple and Fast Multimedia Library (SFML) is a cross-platform software development library designed to provide a simple application programming interface (API) to various multimedia components in computers. It is written in C++ with bindings available for C, Crystal, D, Euphoria, Go, Java, Julia, .NET, Nim, OCaml, Python, Ruby, and Rust. Experimental mobile ports were made available for Android and iOS with the release of SFML 2.2.

SFML handles creating and input to windows, and creating and managing OpenGL contexts. It also provides a graphics module for simple hardware acceleration of 2D computer graphics which includes text rendering using FreeType, an audio module that uses OpenAL and a networking module for basic Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) communication.

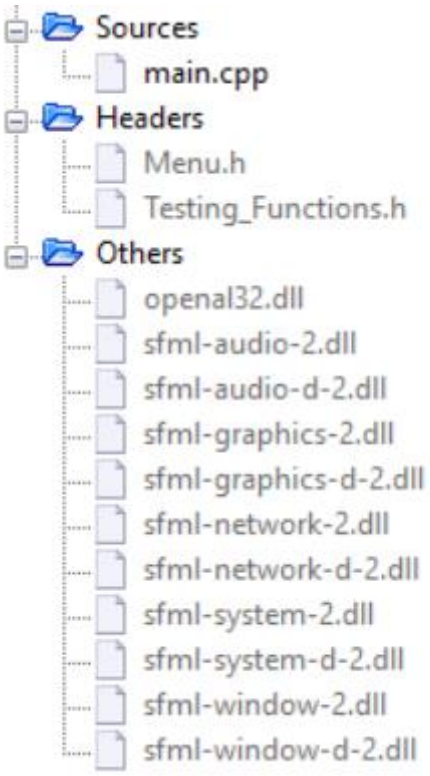
It is free and open-source software provided under the terms of the zlib/png license. It is available on Linux, macOS, Windows and FreeBSD. The first version v1.0 was released on 9 August 2007, the latest version v2.5.1 was released on 15 Oct 2018.

It provides the basic functions on which higher-level software can be built. Add-on libraries exist that provide added support for graphical user interfaces (GUIs), 2D lighting, particle systems and animation, video playback and tilemaps.

SFML consists of various modules:

1. System – vector and Unicode string classes, portable threading and timer facilities
2. Window – window and input device management including support for joysticks, OpenGL context management
3. Graphics – hardware acceleration of 2D graphics including sprites, polygons and text rendering
4. Audio – hardware-accelerated spatialised audio playback and recording
5. Network – TCP and UDP network sockets, data encapsulation facilities, HTTP and FTP classes

File Structure:

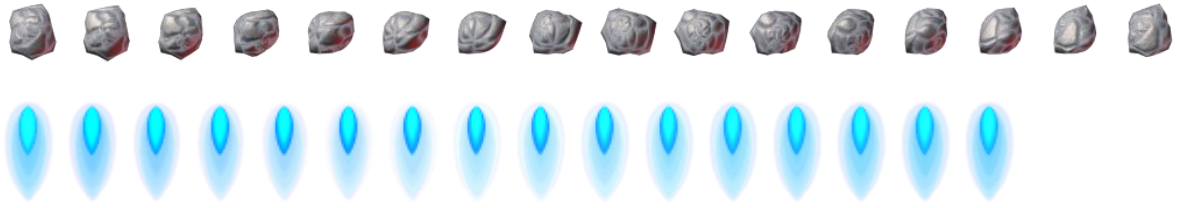


main.cpp contains the main working code

Headers folder contains files which contain that code which was tried out for the game but then later on discarded.

Others folder contains the dynamic linking libraries of SFML

Components used in Game:



Project Details:

Game Mechanics:

1) Movement:

Arrow Keys (Up, Left, Right) are used to move the spaceship around the 2D space. With Up key the spaceship moves forward. Value of thrust variable is changed to 1 which is by default 0 at the start of each frame. With this thrust variable as 1, “dx” and “dy” variables are given certain value according to current angle of the spaceship. Then the values of “dx” and “dy” are used to move the spaceship by updating its position vector value accordingly.

The spaceship keeps moving certain distance even after removing finger from “Up” key to simulate movement in space and this was done by multiplying both “dx” and “dy” by a n.o 0.98 to ensure smooth decay. Update function for all entities is called for each frame.

Asteroids move automatically at a certain angle and speed which varies from one asteroid to another.

Animation class is used to change the frames of the object according to the action the user performs (using keys), and this is done via the functions available within the classes Player, Asteroid and Bullet which are derived from Entity class.

Shooting:

The user presses Spacebar key on the keyboard to fire bullets which on collision with asteroids destroy it and itself.

When a Bullet hits any Asteroid then an explosion occurs which is again nothing but a set of images/frames shown over a certain duration to present the user with a simulation of explosion.

Only one asteroid is destroyed with one bullet.

Continuous press of spacebar doesn't result in a continuous burst of bullets.

Once a bullet goes off the screen then it is destroyed, whereas an asteroid is destroyed only if it hits a bullet or the spaceship.

Random Asteroid Generation:

Random number generator is used to set random angle and speed of newly created asteroid using Asteroid class.

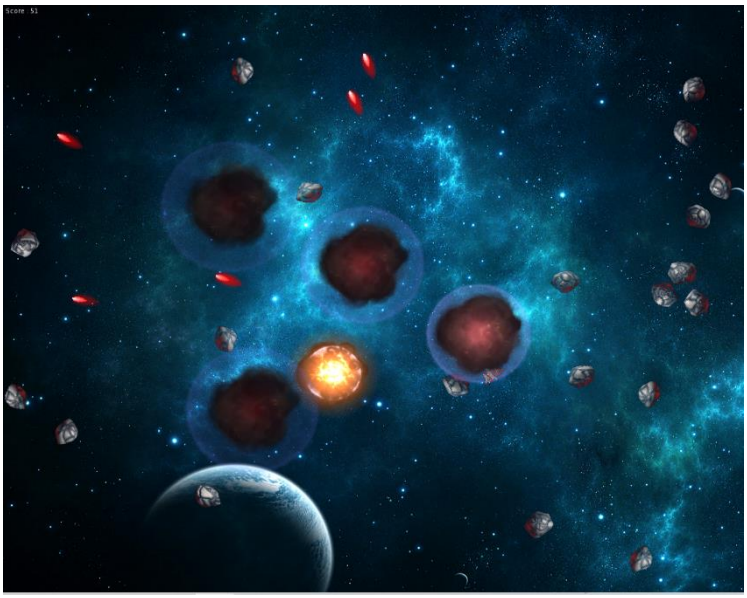
Game Over:

The game ends when the player has used up all of his/her lives.

Snapshots of the Game:

Landing Screen:

GamePlay Screen:



Pause Screen:

Key Implementation Challenges:

This section describes the problems faced during the implementation of this project.

1) Bringing smoothness into the movement of the spaceship was a big challenge.
2) Setting up the different Menu's as part of Graphical User Interface (GUI).
3) Ensuring proper rendering of frames using animation class.
4) Setting up explosion animation.
5) Controlling the flow of game as user presses different buttons.

Results:

We successfully created the 2D Asteroid Game which includes various moving entities namely Asteroid, Spaceship (User), Bullet.

To enhance the environment, collision detection was refined and explosion effects were implemented via a lot of manual tuning of parameters (like tuning the radius of objects).

The randomness of our Asteroid game along with good graphics. Background Image and animation like explosion makes it more challenging yet interesting and enjoyable to play with.

GitHub Link

Conclusion:

SFML is a very good library for making Graphics project with animations.

Animation is a very important part of any computer graphics project as it enables smooth change in the image of the objects thus giving the user sort of moving effect.

OOP is also very important as it reduced the number of lines of code and bugs to a huge extent thereby enhancing the overall development experience and code readability.

Frames are important part for animation. More the number of frames, better is the animation of that particular object.

Some Coding functions

Animation Class Functions:

```
class Animation
{
public:
  float frame, speed;
  Sprite sprite;
  vector<IntRect>frames;

 Animation(Texture &t, int x, int y, int w, int h, int count, float
 spd)
  {
    speed = spd;
    frame = 0;

    for (int i = 0; i < count; i++)
      frames.pb(IntRect(x + i * w, y, w, h));

    sprite.setTexture(t);
    sprite.setOrigin(w / 2, h / 2);

    sprite.setTextureRect(frames[0]);
  }

  void update()
  {
    frame += speed;
    if (frame >= frames.size()) frame -= frames.size();
    if (frames.size() > 0)
sprite.setTextureRect( frames[ (int)frame ] );
  }


  int isEnd()
  {
    return frame + speed >= frames.size();
  }
};
```

Update Function of Player class

```
// update function of Player Class
 void update()
  {
    if (thrust)
    {
      dx += cos(angle * degToRad) * 0.5;
      dy += sin(angle * degToRad) * 0.5;
    }
    else
    {
      // Decay the dx and dy parameters
      dx *= 0.98;
      dy *= 0.98;
    }

    int speed = sqrt(pow(dx, 2) + pow(dy, 2));

    float maxspeed = 15;
    if (speed > maxspeed)
    {
      dx *= (maxspeed / speed);
      dy *= (maxspeed / speed);
    }

    x += dx;
    y += dy;

    cout << "speed=" << sqrt(pow(dx, 2) + pow(dy, 2)) << endl;

    if (x > W)x = 0;  if (x < 0)x = W;
    if (y > H)y = 0;  if (y < 0)y = H;
}
```

Update Function of Bullet class

```
// update function of bullet class
 void update()
 {
    dx = cos(angle * degToRad) * 6;
    dy = sin(angle * degToRad) * 6;

    x += dx;
    y += dy;

    // out of bound then life = 0;
    if (x > W or y > H or x < 0 or y < 0)
      life = 0;
 }
```

Collision Detection Function:

```
// collision detection function:
int collided(Entity* a, Entity* b)
{
  // collision is when the distance between the two entities
  // is <= radii of the two entities
  float dist1 = sqrt( pow(a->x - b->x, 2) + pow(a->y - b->y, 2) ) ;
  float dist2 = a->radius + b->radius;

  return dist1 <= dist2 ;

}
```

References:

1. https://www.sfml-dev.org/tutorials/2.5/start-cb.php
2. https://www.sfml-dev.org/tutorials/2.5/window-window.php
3. https://www.sfml-dev.org/tutorials/2.5/window-events.php
4. https://www.sfml-dev.org/tutorials/2.5/window-inputs.php
5. https://www.sfml-dev.org/tutorials/2.5/graphics-draw.php
6. https://www.sfml-dev.org/tutorials/2.5/graphics-sprite.php
7. https://www.sfml-dev.org/tutorials/2.5/graphics-text.php
8. https://gamefromscratch.com/sfml-c-tutorial-spritesheets-and-animation/
9. http://gamecodeschool.com/sfml/building-your-first-sfml-game-project/
10. https://gamefromscratch.com/sfml-c-tutorial-series/
11. https://stackoverflow.com/
12. https://en.wikipedia.org/wiki/Asteroids_(video_game)#cite_note-IGNReviewSA&MC-30
13. https://en.wikipedia.org/wiki/Asteroids_(video_game)#cite_note-31
14. https://en.wikipedia.org/wiki/Asteroids_(video_game)#cite_note-32
15. https://en.wikipedia.org/wiki/Asteroids_(video_game)#cite_note-33
16. https://en.wikipedia.org/wiki/Asteroids_(video_game)#cite_note-34
17. https://en.wikipedia.org/wiki/Asteroids_(video_game)#cite_note-35
18. https://en.wikipedia.org/wiki/Asteroids_(video_game)#cite_note-40
19. https://en.wikipedia.org/wiki/Asteroids_(video_game)#cite_note-41
20. https://en.wikipedia.org/wiki/Asteroids_(video_game)#cite_note-43