Spike: 16
Title: Collisions

Author: Parth Madhani, 101901858

Goals / deliverables:

- Code
- A developer familiar with simple box and circle based collision test techniques suitable for use in 2D games and their differences.

Technologies, Tools, and Resources used:

- Visual Studio IDE
- Assorted web sources.
  - YouTube
  - Tutorials

Tasks undertaken:

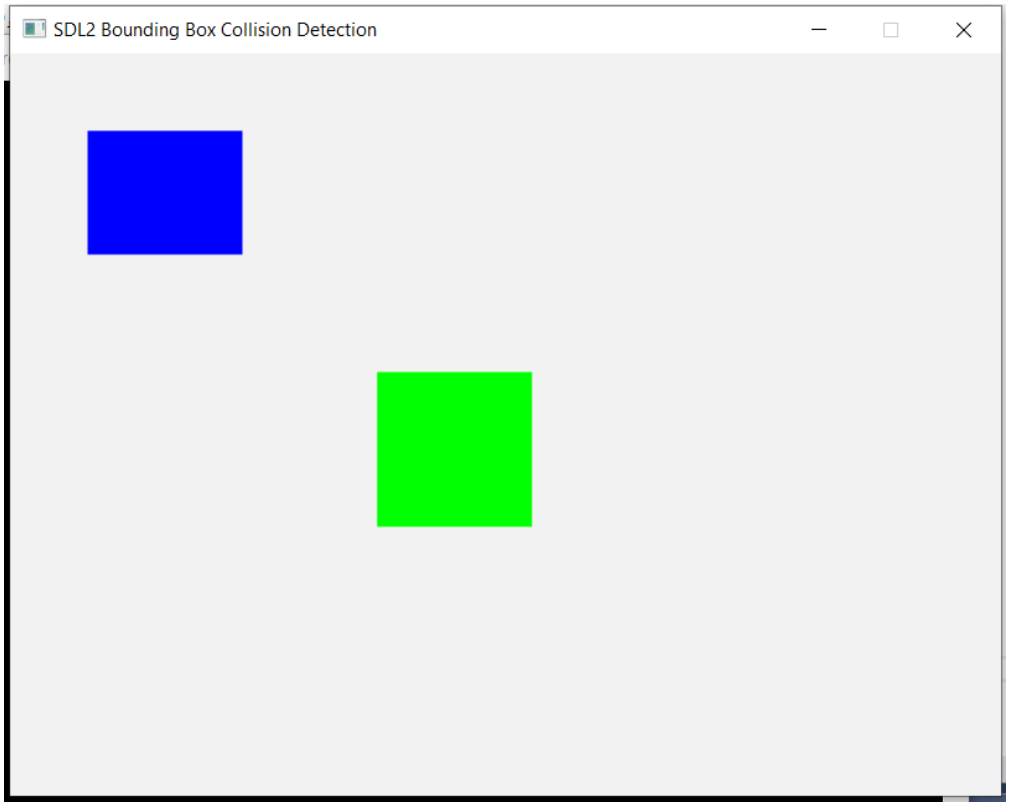
- Research SDL2 framework and how to implement collision with it.
- Implement the code.
- Testing code to ensure it all works the same as before.

What we found out:

We found out how to use SDL2 to check for collisions.
For rectangle collision :

First we draw rectangles on screen then we set keyboard input for one rectangle so it can move using keys.

```
case SDL_KEYDOWN:
    switch (event.key.keysym.sym)
    {
    case SDLK_LEFT:  rect1.x--; break;
    case SDLK_RIGHT: rect1.x++; break;
    case SDLK_UP:    rect1.y--; break;
    case SDLK_DOWN:  rect1.y++; break;
    }
    break;
```



Output with two rectangles drawn.

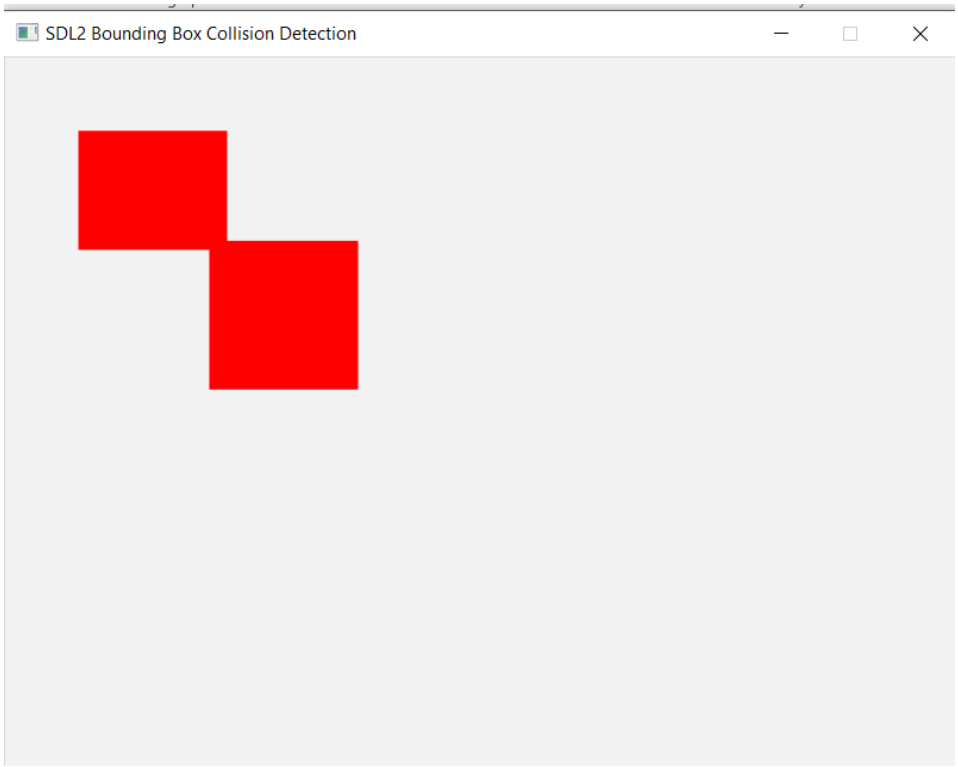
Then we write code to check for collisions.

```
bool checkCollision(SDL_Rect a, SDL_Rect b)
{
    //The sides of the rectangles
    int leftA, leftB;
    int rightA, rightB;
    int topA, topB;
    int bottomA, bottomB;

    //Calculate the sides of rect A
    leftA = a.x;
    rightA = a.x + a.w;
    topA = a.y;
    bottomA = a.y + a.h;

    //Calculate the sides of rect B
    leftB = b.x;
    rightB = b.x + b.w;
    topB = b.y;
    bottomB = b.y + b.h;

    //If any of the sides from A are outside of B
    if (bottomA <= topB)
    {
        return false;
    }

    if (topA >= bottomB)
    {
        return false;
    }

    if (rightA <= leftB)
    {
        return false;
    }

    if (leftA >= rightB)
    {
        return false;
    }

    //If none of the sides from A are outside B
    return true;
```

Here it basically checks if any of the sides collides with other rectangle.

```
SDL_bool collision = SDL_HasIntersection(&rect1, &rect2);

SDL_SetRenderDrawColor(renderer, 242, 242, 242, 255);
SDL_RenderClear(renderer);

if (collision)
    SDL_SetRenderDrawColor(renderer, 255, 0, 0, 255);
else
    SDL_SetRenderDrawColor(renderer, 0, 255, 0, 255);
SDL_RenderFillRect(renderer, &rect1);

if (collision)
    SDL_SetRenderDrawColor(renderer, 255, 0, 0, 255);
else
    SDL_SetRenderDrawColor(renderer, 0, 0, 255, 255);
SDL_RenderFillRect(renderer, &rect2);

SDL_RenderPresent(renderer);
```

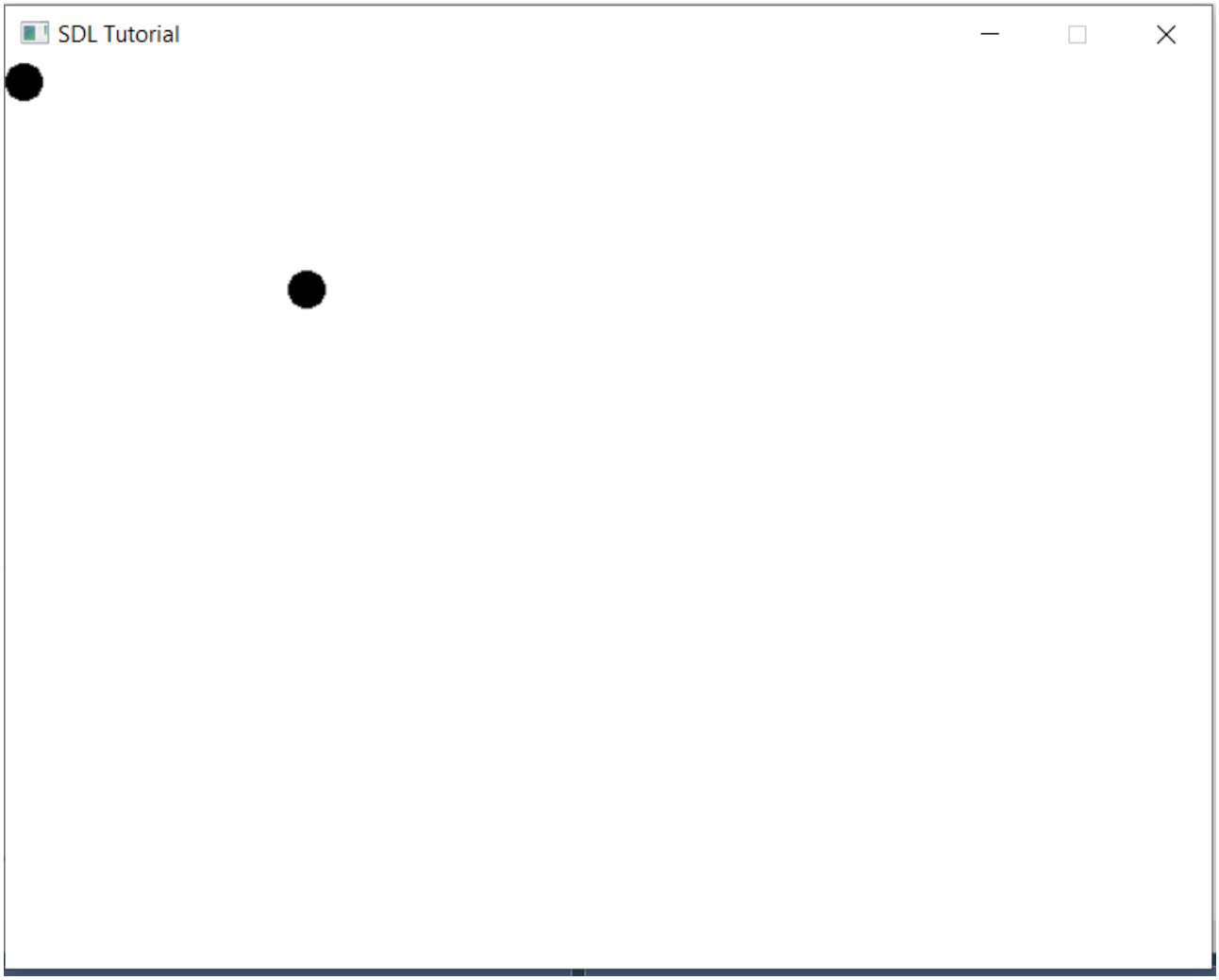
Whenever collision occurs we change color of rectangle to red.



Collision of Rectangles

For Circle :

First we load up two circle images as in SDL2 there is no direct way to draw circle.



Loading up two circles

Next we check for collision between circles, if collided we change their color to red.

```
bool checkCollision(Circle& a, Circle& b)
{
    //Calculate total radius squared
    int totalRadiusSquared = a.r + b.r;
    totalRadiusSquared = totalRadiusSquared * totalRadiusSquared;

    //If the distance between the centers of the circles is less than the sum of their radii
    if (distanceSquared(a.x, a.y, b.x, b.y) < (totalRadiusSquared))
    {
        //The circles have collided
        return true;
    }

    //If not
    return false;
}


double distanceSquared(int x1, int y1, int x2, int y2)
{
    int deltaX = x2 - x1;
    int deltaY = y2 - y1;
    return deltaX * deltaX + deltaY * deltaY;
}
```

Collision code

Output