CS 174A Project Report

Team Slytherin
Link : https://github.com/intro-graphics/team-project-team-slytherin

Name	UID	Github Handle
Minu Jung	304449985	jungm2018
Wilson Lin	904845973	WilsonLin9608
Gawun Kim	305186572	gawun92

1. Description

Game: 3D Snake

The game is an alternate version of the ***Snake*** video game that originated in the 1970s, but with a 3D field of view and additional features such as obstacles and items.

The player controls a snake that moves around in an enclosed space, eating food while avoiding the obstacles. Each time the snake consumes food, its length grows longer.

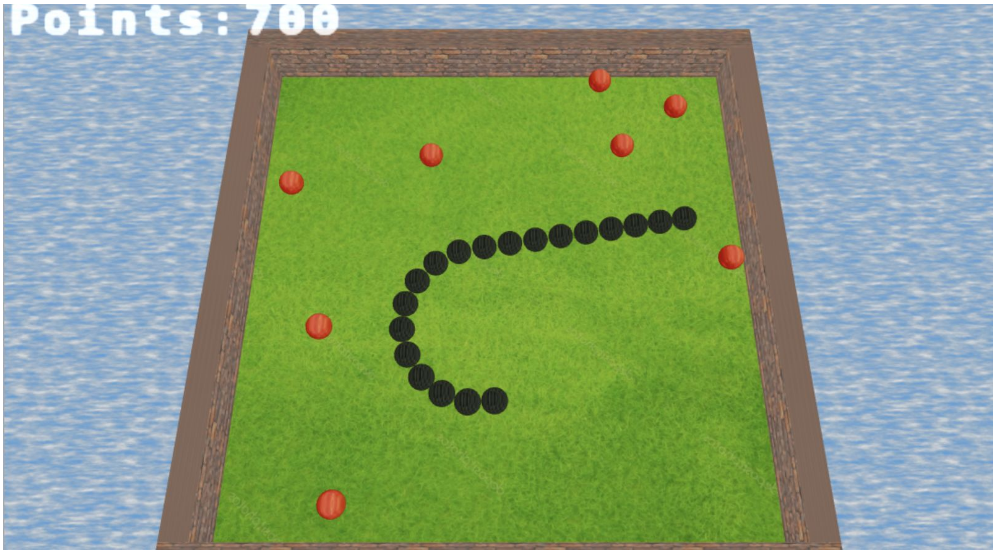
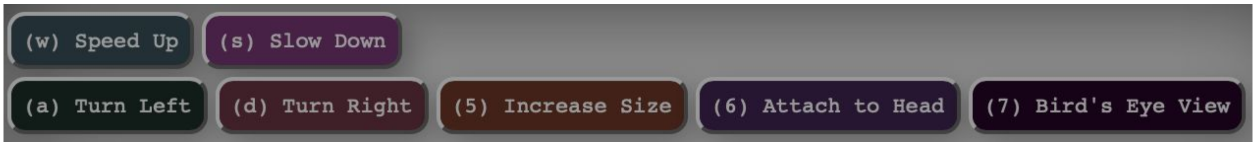


Figure 1. 3D Snake game

1.1 Controls

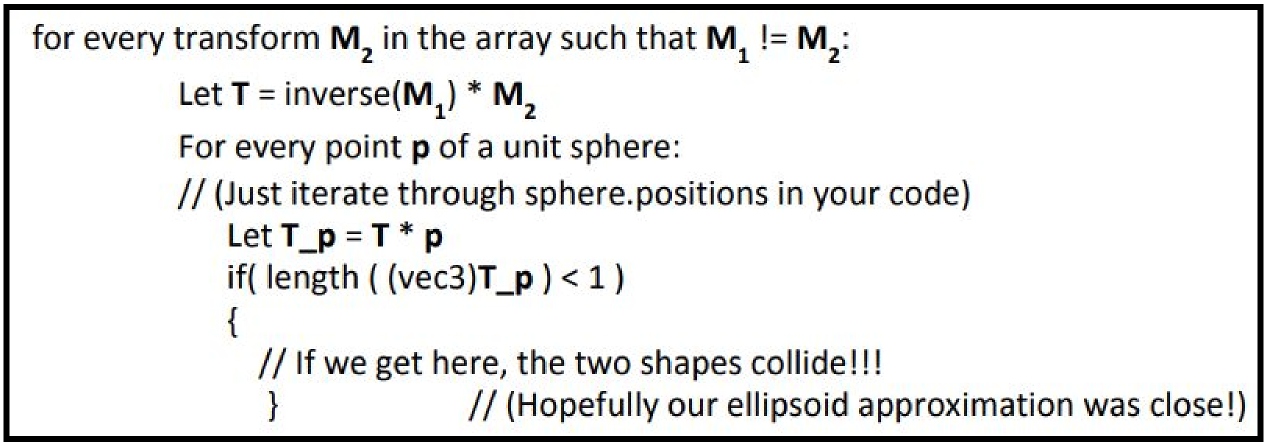


1.2 Game rules

- The obstacle, food, and items are placed randomly in the map
- The player earns points when taking apples (100 points)
- The higher the score, the faster the snake moves
- When the snake is colliding to the wall, the snake dies and the game is over
- When the snake’s head is colliding with its body, the snake will lose some parts of its body and lose points.
- When the snake takes items, it would get the following effects:
    1. Speed up
    2. Longer snake’s body
    3. Extra life

2.1 Features - Collision Detection

The snake will be able to experience collisions with 3 types of objects: Walls, objects (items), and the snake body. Collisions with walls or obstacles result in the loss of a life, with the special case of the snake colliding with itself, upon which the snake length will reduce to length between the head and the collision point. Collisions can be detected by approximating objects’ shape with collision ellipsoids. Each object in the game has their own model transforms, by using the model transforms on a unit sphere we can generate such collision ellipsoids and by comparing each model transforms we can detect whether there is a collision or not. The implementation is similar to the one provided on the slides on week 6, with the only difference being that we only need to check collisions with the head of the snake and when we place a new object in the map, since they are the only events that can cause a collision. However, in order for the approximation to be accurate, each object’s vertices must be centered around the origin and roughly unit length in all directions.



```
for every transform $M_2$ in the array such that $M_1$ != $M_2$:
        Let T = inverse($M_1$) * $M_2$
        For every point p of a unit sphere:
        // (Just iterate through sphere.positions in your code)
            Let T_p = T * p
            if( length ( (vec3)T_p ) < 1 )
            {
              // If we get here, the two shapes collide!!!
              }            // (Hopefully our ellipsoid approximation was close!)
```

Figure 2.1. Collision detection implementation

3, What we have done in the demo

3.1 Structure

Class : “Team_Slytherin_Project”

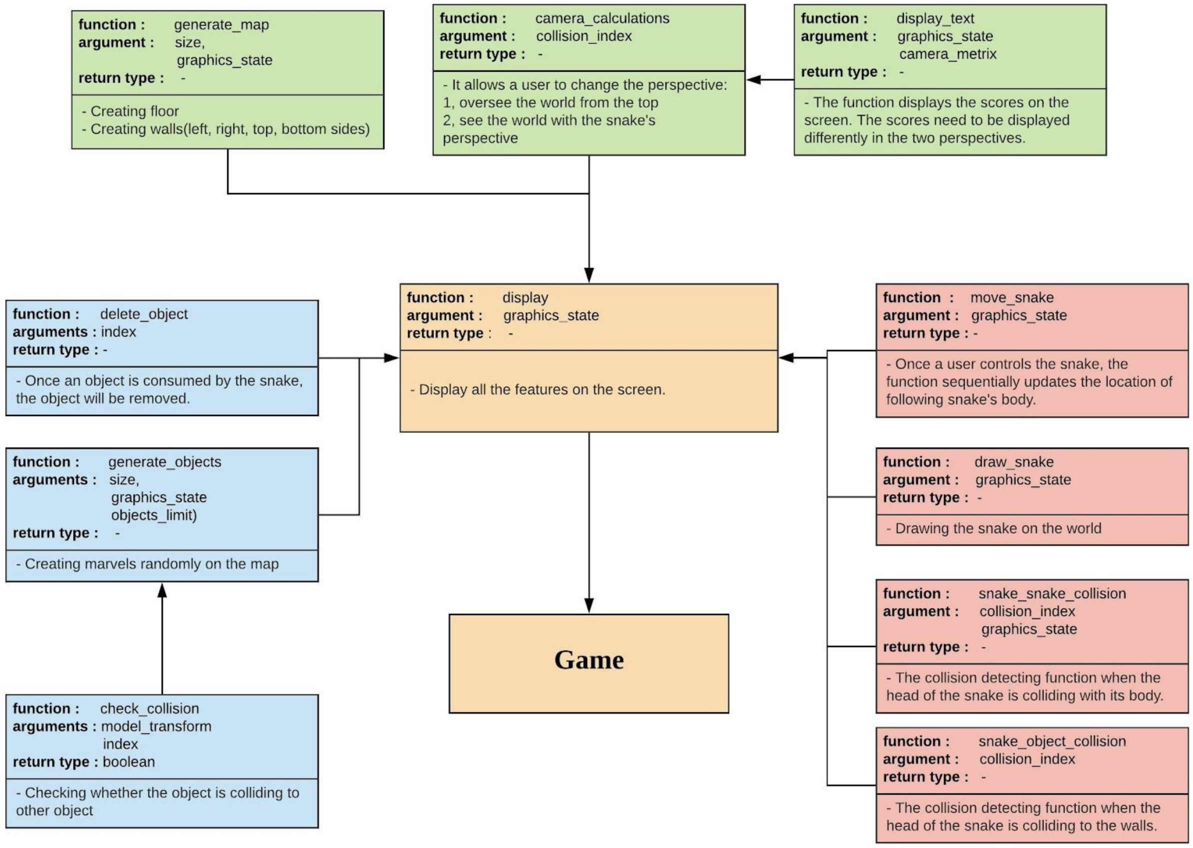

function : generate_map
argument : size, graphics_state
return type : -
- Creating floor
- Creating walls(left, right, top, bottom sides)
function : camera_calculations
argument : collision_index
return type : -
- It allows a user to change the perspective:
1, oversee the world from the top
2, see the world with the snake's perspective
function : display_text
argument : graphics_state camera_metrix
return type : -
- The function displays the scores on the screen. The scores need to be displayed differently in the two perspectives.
function : delete_object
arguments : index
return type : -
- Once an object is consumed by the snake, the object will be removed.
function : display
argument : graphics_state
return type : -
- Display all the features on the screen.
function : move_snake
argument : graphics_state
return type : -
- Once a user controls the snake, the function sequentially updates the location of following snake's body.
function : generate_objects
arguments : size, graphics_state objects_limit)
return type : -
- Creating marvels randomly on the map
function : draw_snake
argument : graphics_state
return type : -
- Drawing the snake on the world
Game
function : snake_snake_collision
argument : collision_index graphics_state
return type : -
- The collision detecting function when the head of the snake is colliding with its body.
function : check_collision
arguments : model_transform index
return type : boolean
- Checking whether the object is colliding to other object
function : snake_object_collision
argument : collision_index
return type : -
- The collision detecting function when the head of the snake is colliding to the walls.

Figure 3.1. Structure of the game

Orange box: it is the main function to draw all objects in the screen.

Blue box: the functions related to creating or removing objects(items) on the map.

Red box: The functions related to the snake’s control, movement, and collision update.

Green box : the functions related to the background and perspective.

3.2 Features

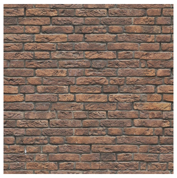
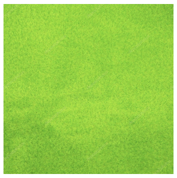
Basic object:

We created the background (world) and the snake objects. The world is the multiple combination of cubes and the board; the snake is the series of multiple marbles.

Behaviors:

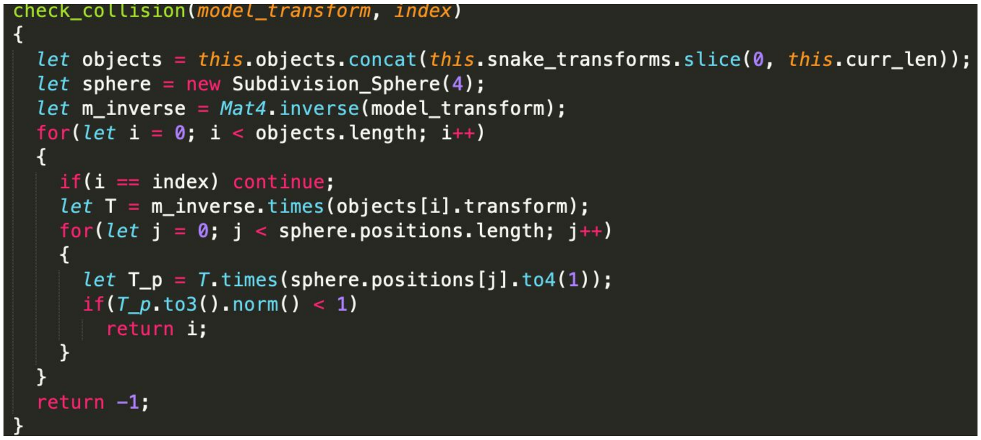
The snake is moving straight as a default behavior and can go left or right. When changing the direction, it turns around as a circular shape. The body of the snake is following the head of the snake and each location of each marble is updated to the location of the front marbles. The items are created in the world randomly and it is consumable by the snake. If it is consumed, a new item will be created in the random location and it always keeps the constant number of items in the map. When the snake is colliding into the wall, the snake is considered as dead and the game is over.

Texture:

1	**Wall**		4	**Apple**	
2	**Sky**		5	**Snake**	
3	**Ground**				

Collision:

Based on the return value from the function(check_collision), it detects whether the collision between objects exists or not; the collision detection algorithm has been used. As a big picture, the collision can be categorized into the three: snake with wall, snake with object, snake with snake. All objects are saved into an array with the order (Wall, Apple, Snake body) and the type of the objects can be specified by calling (object.type). It allows not only detecting the object collision but also it helps recognizing which objects are colliding with each other.



```
check_collision(model_transform, index)
{
  let objects = this.objects.concat(this.snake_transforms.slice(0, this.curr_len));
  let sphere = new Subdivision_Sphere(4);
  let m_inverse = Mat4.inverse(model_transform);
  for(let i = 0; i < objects.length; i++)
  {
    if(i == index) continue;
    let T = m_inverse.times(objects[i].transform);
    for(let j = 0; j < sphere.positions.length; j++)
    {
      let T_p = T.times(sphere.positions[j].to4(1));
      if(T_p.to3().norm() < 1)
        return i;
    }
  }
  return -1;
}
```

Figure 3.2. Collision Detection implementation

Controls:

Speed up(‘w’) - the snake is moving faster

Slow down(‘s’) - the snake is moving slower

Left(‘a’) - the snake moves its direction left side as a circular motion

Right(‘d’) - the snake moves its direction right side as a circular motion

Increase size(‘5’) - for one hit, the snake gains one marble on its tail.

Attach to Head(‘6’) - the user can change the perspective of the playing scene to snake perspective. With the perspective change, the score will be displayed differently; right above the snake head.

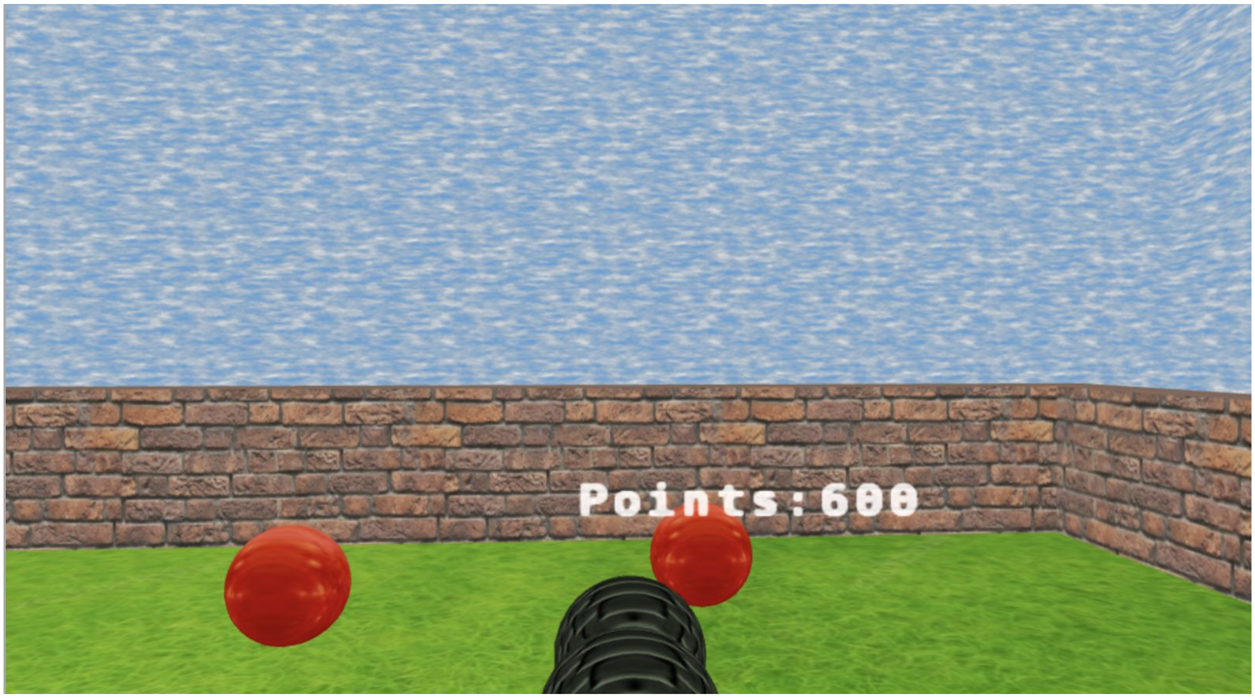


Figure 3.3. Attach to Head (perspective 1)

Bird’s Eye View(‘7’) - it is a default perspective and it allows a user to see all the world and a snake from above. With the perspective change, the score will be displayed differently; located to the left above of the game screen.

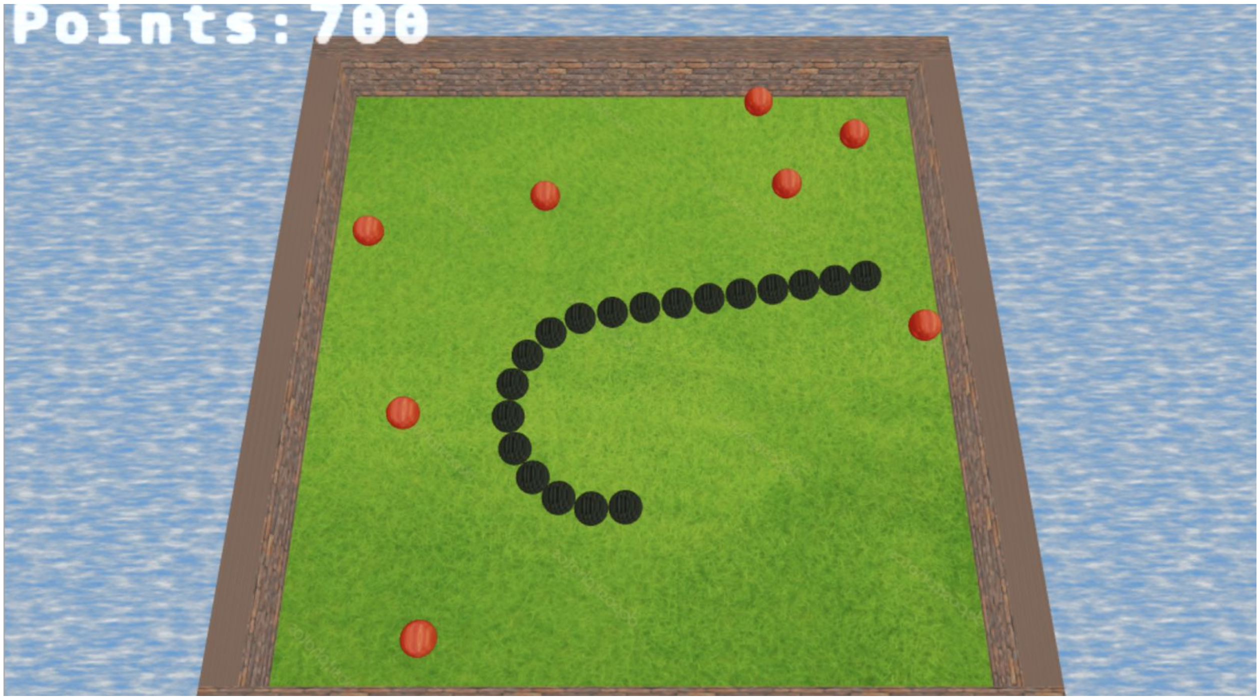


Figure 3.4 Bird’s Eye View (perspective 2)

4, Interesting things

1, screen shaking - When the snake is colliding with its body, the game screen is shaking left and right for one second.

2, rat object - due to heavy workload, the function is not completed and commented out.

[completed one] The rat runs away from the snake. When the euclidean distance is within 20, the rat is reacting to it and if not, it does not react.

[incompleted one] When the rat is moving away from the snake, the rat can encounter the apple objects and its behavior(direction) is complicated and it took a lot of time to make it. For this reason, we decided to comment it out.

5, Contribution

Commits:

Format : https://github.com/intro-graphics/team-project-team-slytherin/commit/[number]

Wilson

8dc3f7f6372304edb883bb66aadc2cdc44f10bcc

D61b25bfe34d1e0f5b8f25751152e1a6ee141d2d

95a4502c0da6a7f1663a532db8b9c7a763f8ebd7

3a9f0a2004544bdd14539de159e7ce97704fc162

05a5be1fbbab71f012499493d4429043f2b97d07

5cf44e1b5b1070bf8bcaa6908206ed372eb20a79

5fc2083d3e32f1c4a5ea60f78884c9ce0de4446a

7e2e8b326c83c4e678d58cde0a51923af060ef27

002e649f1e6e13e519c4e9d7bbf00e73a0739cf3

4c35758458861c59aa87c9f7030ec6055f4631d9

B9ed41946b9e33b2c84308210e3142f7b444a766

C730ec6372e3497dec79ca9f2bf4004e56536617

08261fea6d99ed4261c5e58a4624fdb40c37fcd6

8929d0efc63739c5eb7daf57e79e58c3e6531e86

2c0855a0091fb66b97740f1f9f051d53c9c95c0c

3098ae91f1f90e48a2ea7b7b390932a8ccff6080

Aa9745e262091e34f42416b2cfa5732b1b9cbec1

1c9f8393235533176440 0ed55de9879b19e16b07

16b2aa7ffb188623778a22884bf435febed5edf2

93505180993ddd1270b03c0271628fc12961a68e

537ae53a9a86697314f049773932550e4543ac85

F2304009bc54caa2c3e7af56d6d5aa744c760e6e

5ebccd33f5ed6c7a0aa251232ac14052afb0617e

Ef831e33f2e130f73d17c94f5f464cf033171e72

Eb76a40e8ffa4bf94a155aaf592909c67ee365ac

Cd5c3cd5a281a6b6333345ac878162b9d97e53e9

F23cdf3b4185b0b6b44c2688519d90c4c091b967

1eff314685bbaead1cf75179a802e844fc8e321c

5d7962245cb2ea28fb77d57b4de64d0a73916255

B3ab00191e1134681bb369f84a30f8da2cf99700

0e48e731f73ac4173f203578f8b9b02c0203c293

a3734e92363029d42204ecc9edc50abee880eb63

E33a054b8c7717e0c9c13b4750d85751155f2e36

A63979d69a2aeac8f4f0a0f70dfad92d9a32a0c6

faf6a96cafd2659b376039288e8cd0152f14e316

Minu Jung

f460374a3bf415611cdd34448189b185a084da8c

681f77a5a3f493cd7e3cb48bfa382612ea63fcf8

0acc2c559aab2830397700aafa407125cf61155e

E278188d4f726397d21a59f62d0724b921b9360f

0a6dd2c555332dd0a49711e085aab39bd3fa638c

99a7b469e3863e6698c59f1708f878d825d82111

E549fa5c25f4ca709fd64b439a19c581690ead48

42b0152495890a30f1c38319487ca3c638a5ad8a

Bc883a030aaadf4a38501b2b2cf9dc74d65e3eef

9bc586f2ebbe1dde6bed609b6bdb71069c515c7d

e4ceb0d9a928bbbdb9544a3b9b593d888aeaa655

Gawun Kim

Afd2544fcff8f6887ab32db3cd980204b7037c2d

0e6e713173a907eeea23ee468f7a362d627ead25

Ff673928614c406d574e3460d61071d12333f891

4794af42e125cd891d5b1c8118f16b66651888e5

E4d5b6a5996e8f1a4b47e0049a07d20c29cb84f4

7c706058422607453c9ada2ef31efa1b3888ce8f