Bachelors Thesis

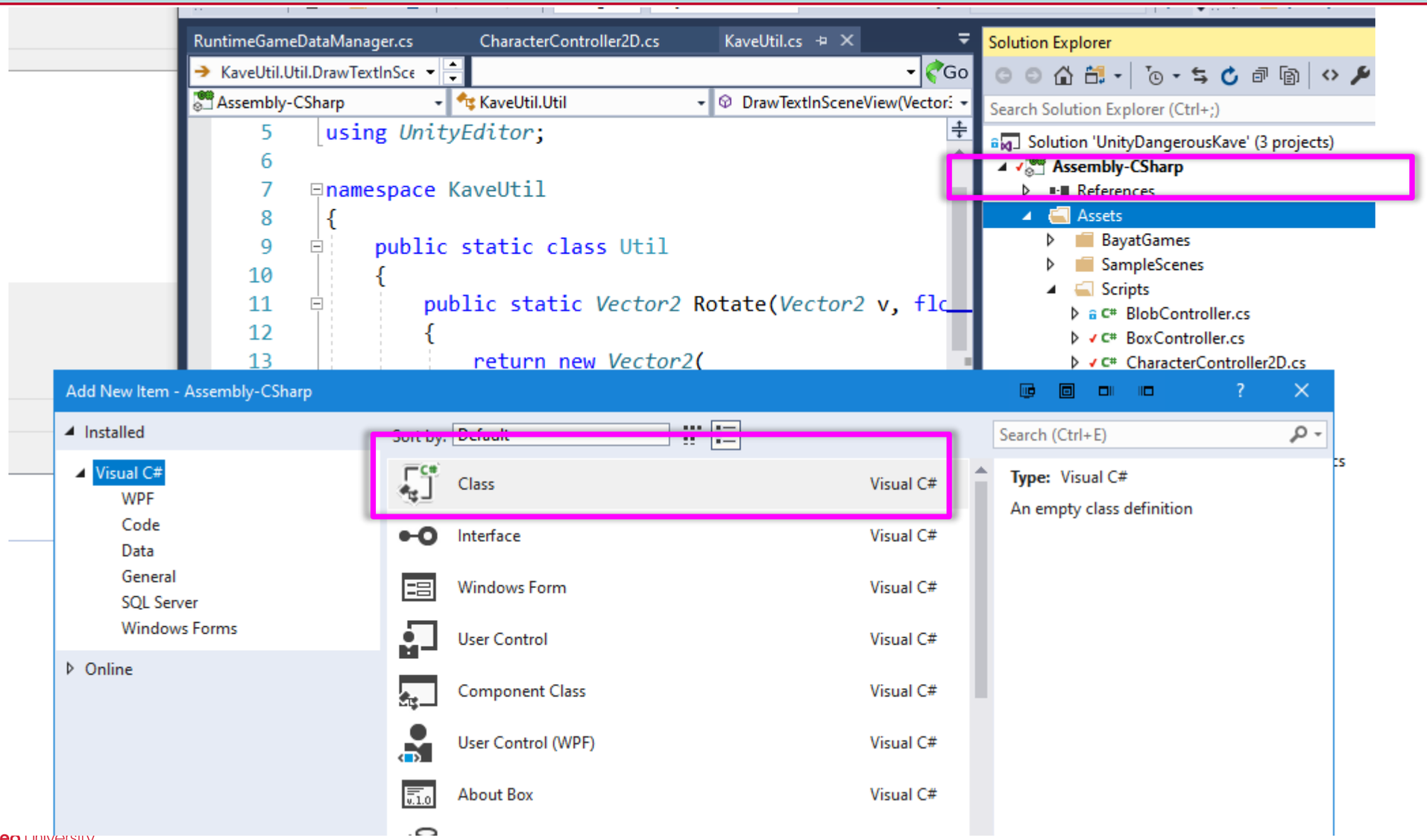
Dangerous Kave Week03

jintaeks@dongseo.ac.kr

March 31, 2020

DSU Dongseo University 동서대학교

namespace KaveUtil



```
namespace KaveUtil{
    public static class Util    {
        public static Vector2 Rotate(Vector2 v, float delta)        {
            return new Vector2(
                v.x * Mathf.Cos(delta) - v.y * Mathf.Sin(delta),
                v.x * Mathf.Sin(delta) + v.y * Mathf.Cos(delta)
            );
        }

        public static void DrawTextInSceneView(Vector3 worldPos, string text, Color? colour =
null)        {
            UnityEditor.Handles.BeginGUI();
            {
                if (colour.HasValue) GUI.color = colour.Value;
                SceneView view = UnityEditor.SceneView.currentDrawingSceneView;
                Vector3 viewPos = view.camera.WorldToScreenPoint(worldPos);
                viewPos.y += 18;
                Vector2 screenPos = new Vector2(viewPos.x, -viewPos.y + view.position.height);
                GUI.Box(new Rect(screenPos, new Vector2(128, 64)), text);
            }
            UnityEditor.Handles.EndGUI();
        }
    }//class Util
```

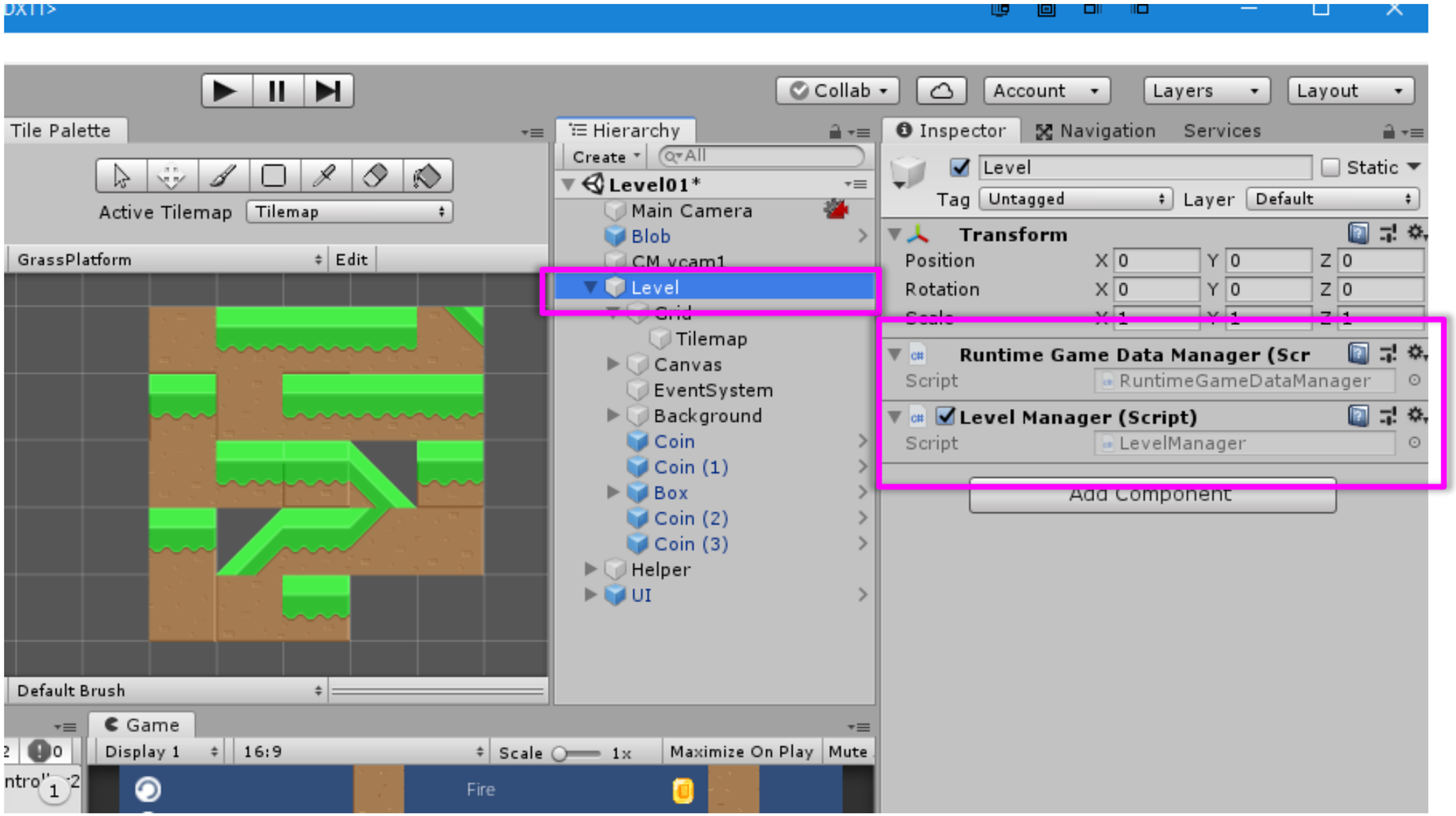
CircularQueue⟨T⟩

```
class CircularQueue<T>    {
        private T[] element;
        private int front;
        private int rear;
        private int max;
        private int count;

        public CircularQueue(int size)        {
            element = new T[size];
            front = 0;
            //rear = -1; // not good
            rear = 0; // use half closed interval
            max = size;
            count = 0;
        }
        public int GetCount()        {
            return count;
        }
```

```
public bool Insert(T item, bool bAutoDelete = true)        {
    if (count == max)            {
        if (bAutoDelete == false)
            return false;

        Delete();
    }

    element[rear] = item;
    rear = (rear + 1) % max;
    count++;
    return true;
}
public bool Delete()        {
    if (count == 0)             {
        return false; // empty error
    }

    front = (front + 1) % max;
    count--;
    return true;
}
```

```
public bool GetFront(out T e)         {
            if (count <= 0)             {
                e = default(T);
                return false;
            }
            e = element[front];
            return true;
        }
        public bool GetRear(out T e)         {
            if (count <= 0)             {
                e = default(T);
                return false;
            }
            e = element[rear];
            return true;
        }
    }// class CircularQueue<T>
```

Pseudo Singleton



```
public class RuntimeGameDataManager : MonoBehaviour {
    // Now we are using the pseudo singleton.
    // 20200328_jintaeks
    //static private RuntimeGameDataManager instance;
    //static public RuntimeGameDataManager Instance
    //{
    //    get { return instance; }
    //}
    private static int _dataStamp = 0;
    public static int DataStamp     {
        get { return _dataStamp; }
    }
    private static int _coinCounter = 0;
    //GameObject _player;

    private void Awake()     {
        // data initialization can be placed here
        // 20200328_jintaeks

        //if (instance == null)
        //    instance = this;
        //_player = GameObject.FindGameObjectWithTag("Player");
        //CharacterController2D cc2d = _player.GetComponent<CharacterController2D>();
        //cc2d.OnCollision += OnCollisionCallback;
    }
```

```
static void UpdateDataStamp()
    {
        ++_dataStamp;
    }

    public static int GetCoinCounter()
    {
        return _coinCounter;
    }

    public static void AddCoinCounter(int c)
    {
        _coinCounter += c;
        UpdateDataStamp();
    }
}//public class RuntimeGameDataManager : Mo
```

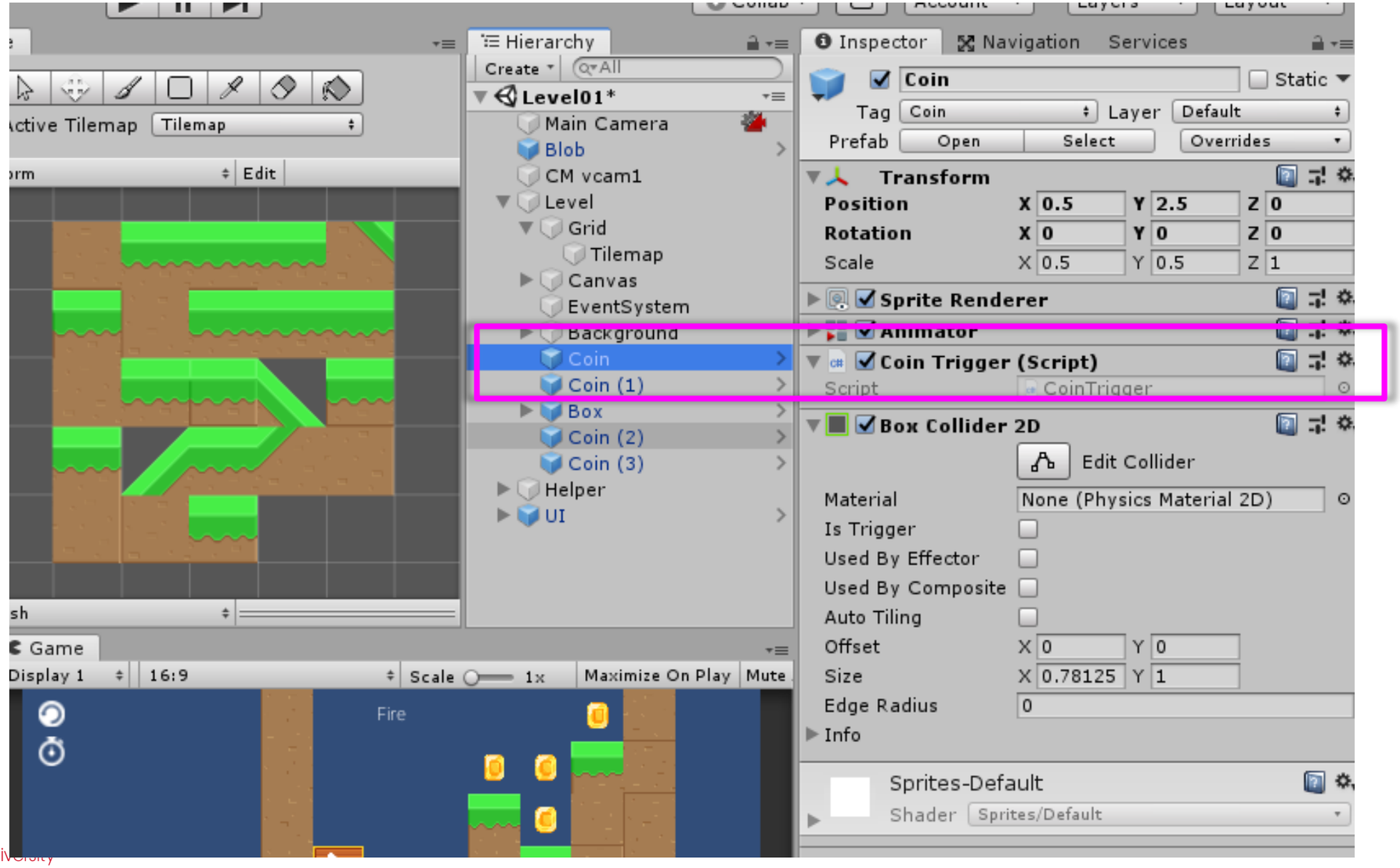
```
 void OnCollisionCallback(Collider2D coll2d,
ColliderDistance2D collDist2d)
    {
        if (coll2d.gameObject.CompareTag("Coin"))
        {
            Destroy(coll2d.gameObject);

RuntimeGameDataManager.AddCoinCounter(1);
        }
    }
```

LevelManager

```
public class LevelManager : MonoBehaviour
{
    static private TilemapCollider2D _tilemap2d;

    void Awake()
    {
        Transform tilemap = transform.Find("Grid/Tilemap");
        _tilemap2d = tilemap.gameObject.GetComponent<TilemapCollider2D>();
    }

    public static bool IsOverlapWithTilemap(Vector2 p)
    {
        if( _tilemap2d != null )
            return _tilemap2d.OverlapPoint(p);

        return false;
    }
}
```

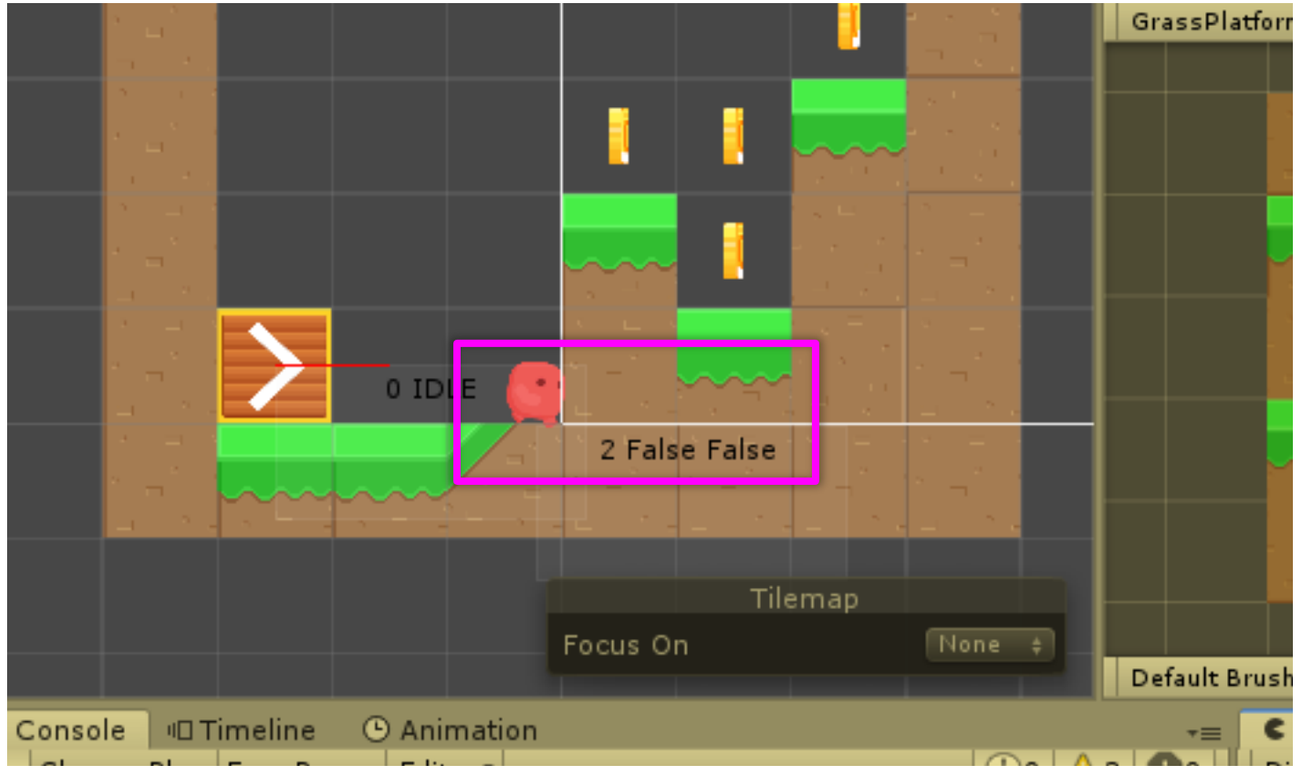
```
Vector2 groundPos = transform.position;
        groundPos.y -= 0.02f;
        if
(LevelManager.IsOverlapWithTilemap(groundPos))
            _isGrounded = true;

        if (isJumping != _isJumping)
```

CoinTrigger



```
public class CoinTrigger : MonoBehaviour {
    private GameObject _player;

    private void Awake()     {
        _player = GameObject.FindGameObjectWithTag("Player");
        CharacterController2D cc2d = _player.GetComponent<CharacterController2D>();
        cc2d.OnCollision += OnCollisionCallback;
    }

    private void OnTriggerEnter2D(Collider2D collision)     {
        // will not be called, because collision response is processed in the
'CharacterController2D'
        // 20200328
    }

    void OnCollisionCallback(Collider2D coll2d, ColliderDistance2D collDist2d)     {
        if (coll2d.gameObject.CompareTag("Coin"))
        {
            Destroy(coll2d.gameObject);
            RuntimeGameDataManager.AddCoinCounter(1);
        }
    }
}
```

CharacterController2D: '_isGrounded' problem



BoxCastAll() doesn't detect Tilemap Collider 2D

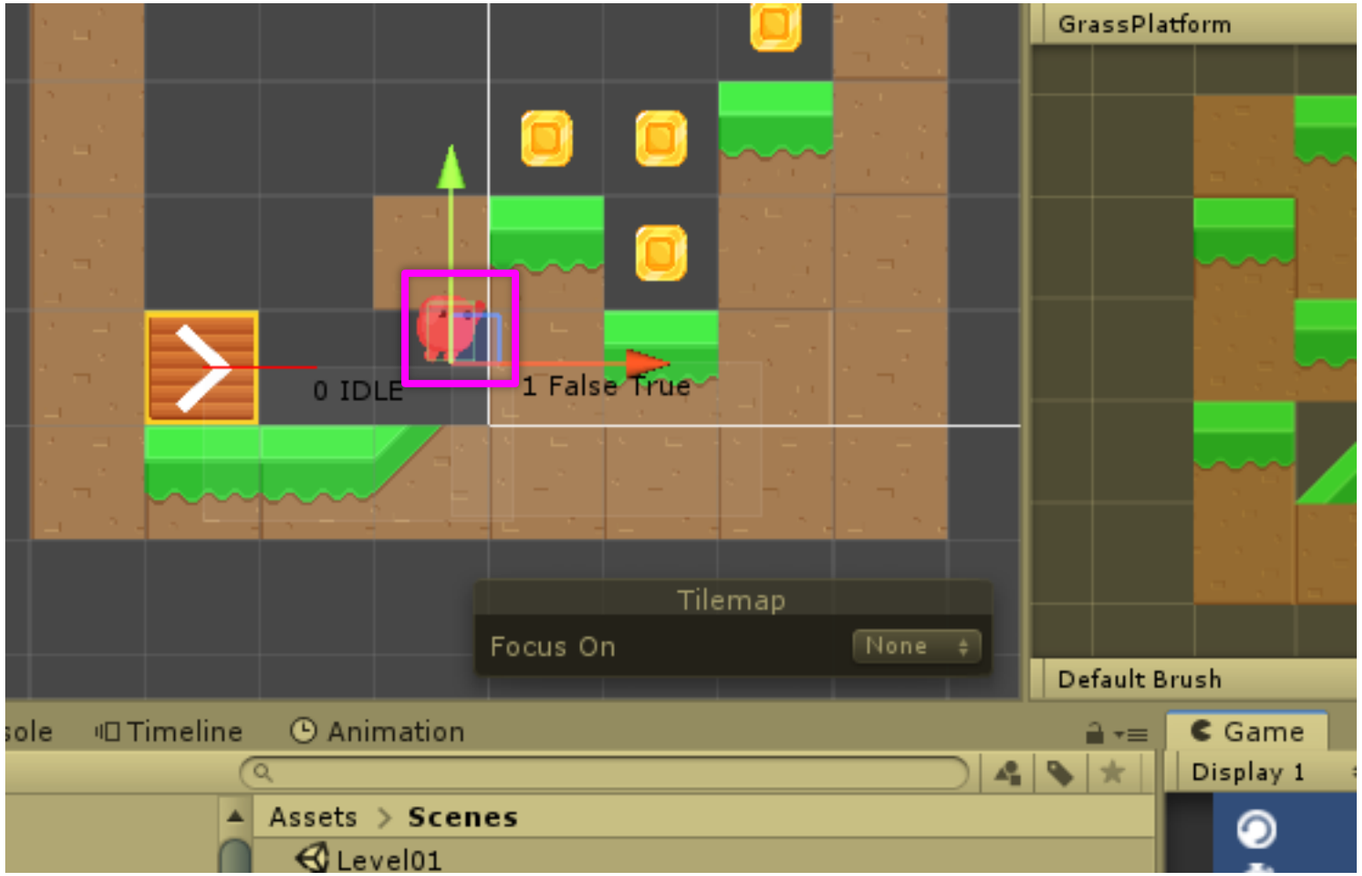
- Contents

(contents)

```
RaycastHit2D[] hit2d = Physic   BoxCastAll(   nsform.position,
_boxCollider.size, 0, Vector2.up)
        float fMaxDistSoFar = 0.0f;
        foreach (RaycastHit2D hit in    )
        {
            if (hit.collider == _b   llic
                continue;
            if (hit.distance   axDistSoFar)
                fMaxDistSoFar   hit.distance;
        }
        _castDistance = fMaxDistSoFar;
```

```
bool isGrounded = false;

// Retrieve all colliders we have intersected after velocity has been applied.
Collider2D[] hits = Physics2D.OverlapBoxAll(transform.position, _boxCollider.size, 0);
_hitCount = hits.Length;

foreach (Collider2D hit in hits)        {

    if (colliderDistance.isOverlapped)            {

        // If we intersect an object beneath us, set grounded to true.
        if (Vector2.Angle(colliderDistance.normal, Vector2.up) < 90 && _velocity.y <=
0)
        {
            isGrounded = true;
        }
    }
}

_isGrounded = isGrounded;

Vector2 groundPos = transform.position;
groundPos.y -= 0.02f;
if (LevelManager.IsOverlapWithTilemap(groundPos))
    _isGrounded = true;
```

Ceil penetration problem

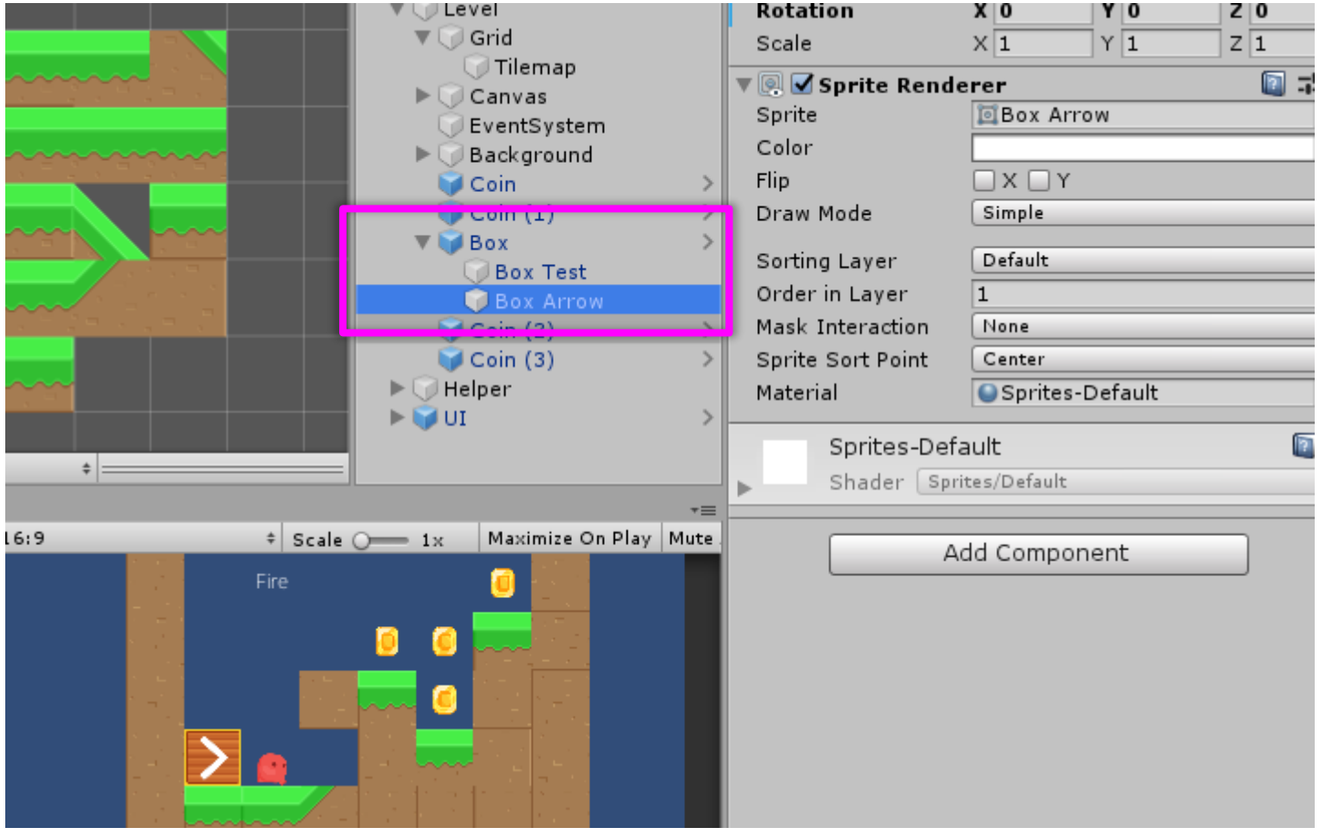


```
foreach (Collider2D hit in hits)        {

    if (colliderDistance.isOverlapped)              {
        if (hit.gameObject.CompareTag("Coin") == false)                 {
            transform.Translate(colliderDistance.pointA - colliderDistance.pointB);
        }//if

        // If we intersect an object beneath us, set grounded to true.
        if (Vector2.Angle(colliderDistance.normal, Vector2.up) < 90 && _velocity.y <=
0)          {
            isGrounded = true;
        }
        if (Vector2.Angle(colliderDistance.normal, Vector2.down) < 90 && _velocity.y >
0)          {
            _isCeilCollision = true;
            _velocity.y = 0;
        }
    }
}

_isGrounded = isGrounded;
```

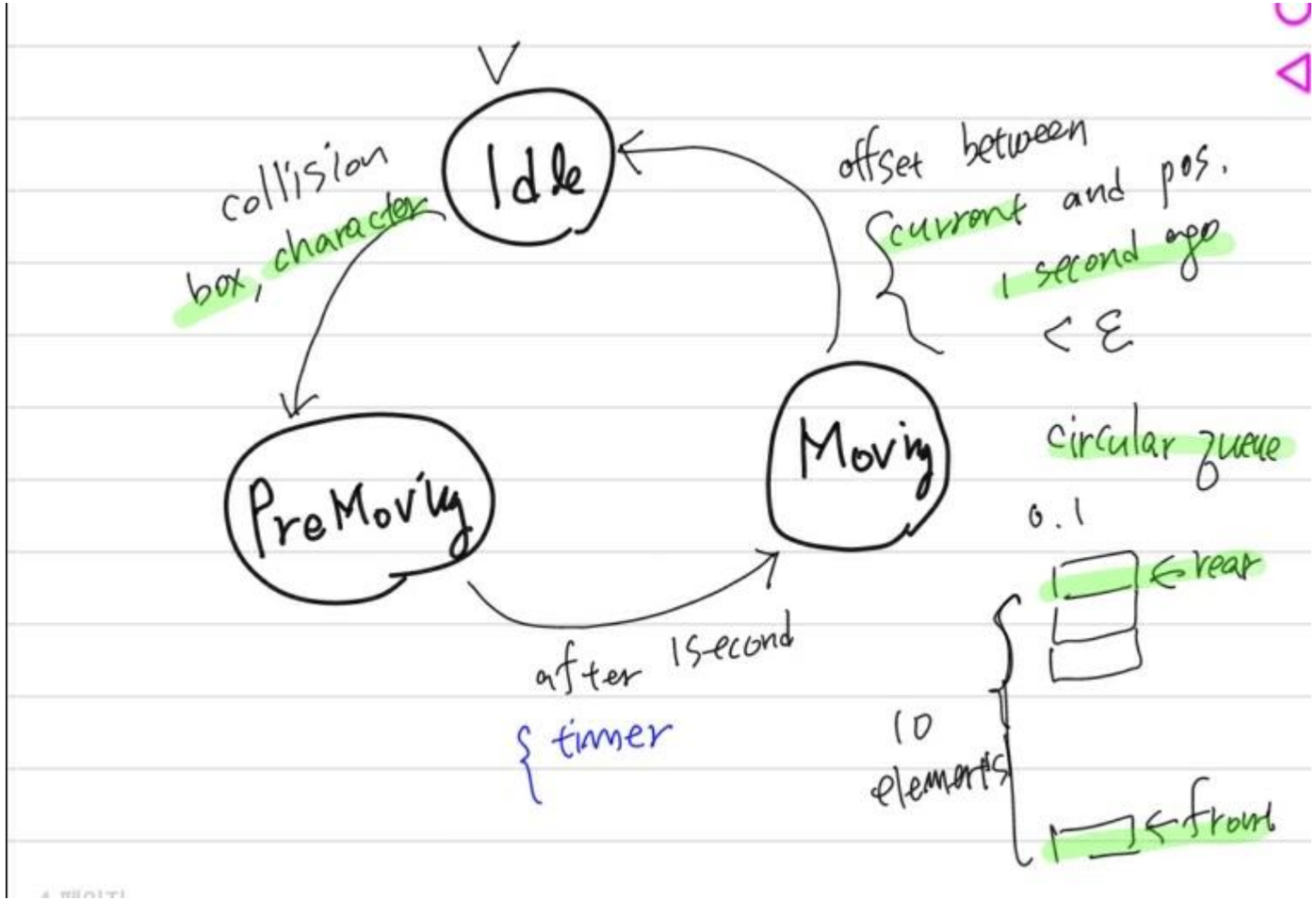
Coin Collision Problem

```
foreach (Collider2D hit in hits)        {

        if (colliderDistance.isOverlapped)              {
            if (hit.gameObject.CompareTag("Coin") == false)                 {
                transform.Translate(colliderDistance.pointA - colliderDistance.pointB);
            }//if

            // If we intersect an object beneath us, set grounded to true.
            if (Vector2.Angle(colliderDistance.normal, Vector2.up) < 90 && _velocity.y <=
0)              {
                isGrounded = true;
            }
            if (Vector2.Angle(colliderDistance.normal, Vector2.down) < 90 && _velocity.y >
0)              {
                _isCeilCollision = true;
                _velocity.y = 0;
            }
        }
    }

    _isGrounded = isGrounded;
```

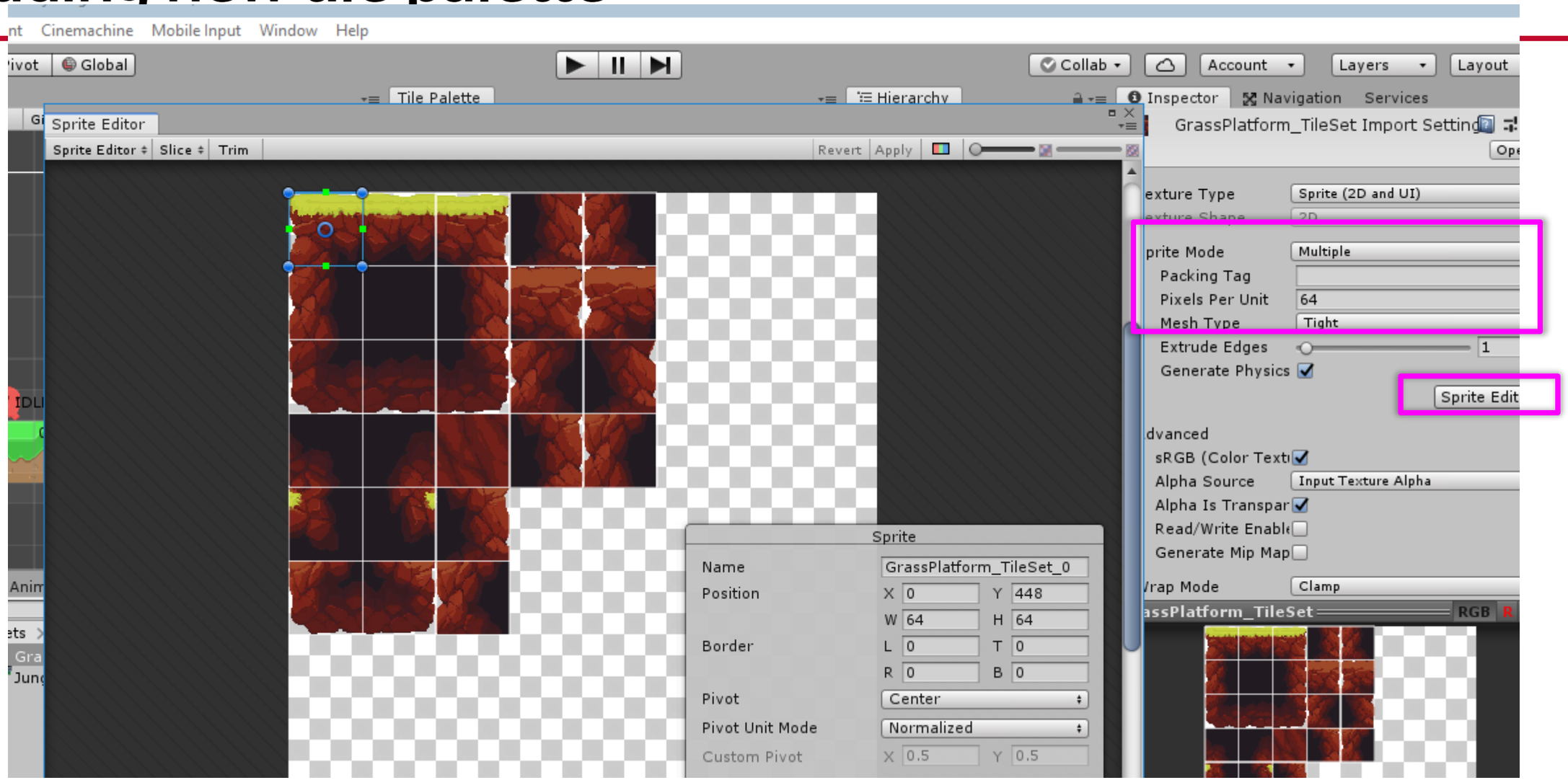
Box Mechanic

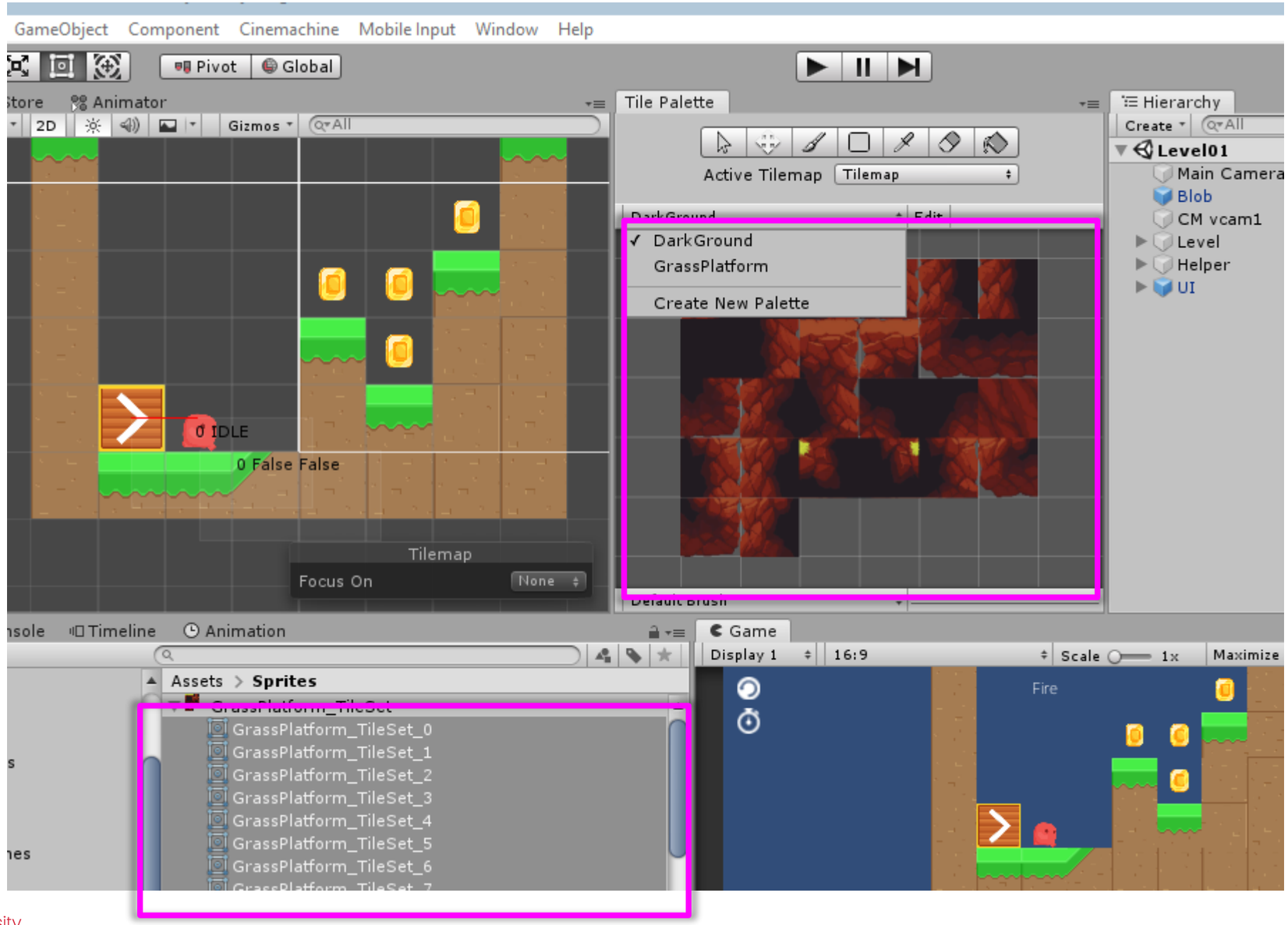


Demo: State Machine of Box Mechanic



Adding new tile palette





MY **BRIGHT** FUTURE

DSU Dongseo University 동서대학교