DIVISION OF
DIGITAL CONTENTS
DONGSEO UNIVERSITY

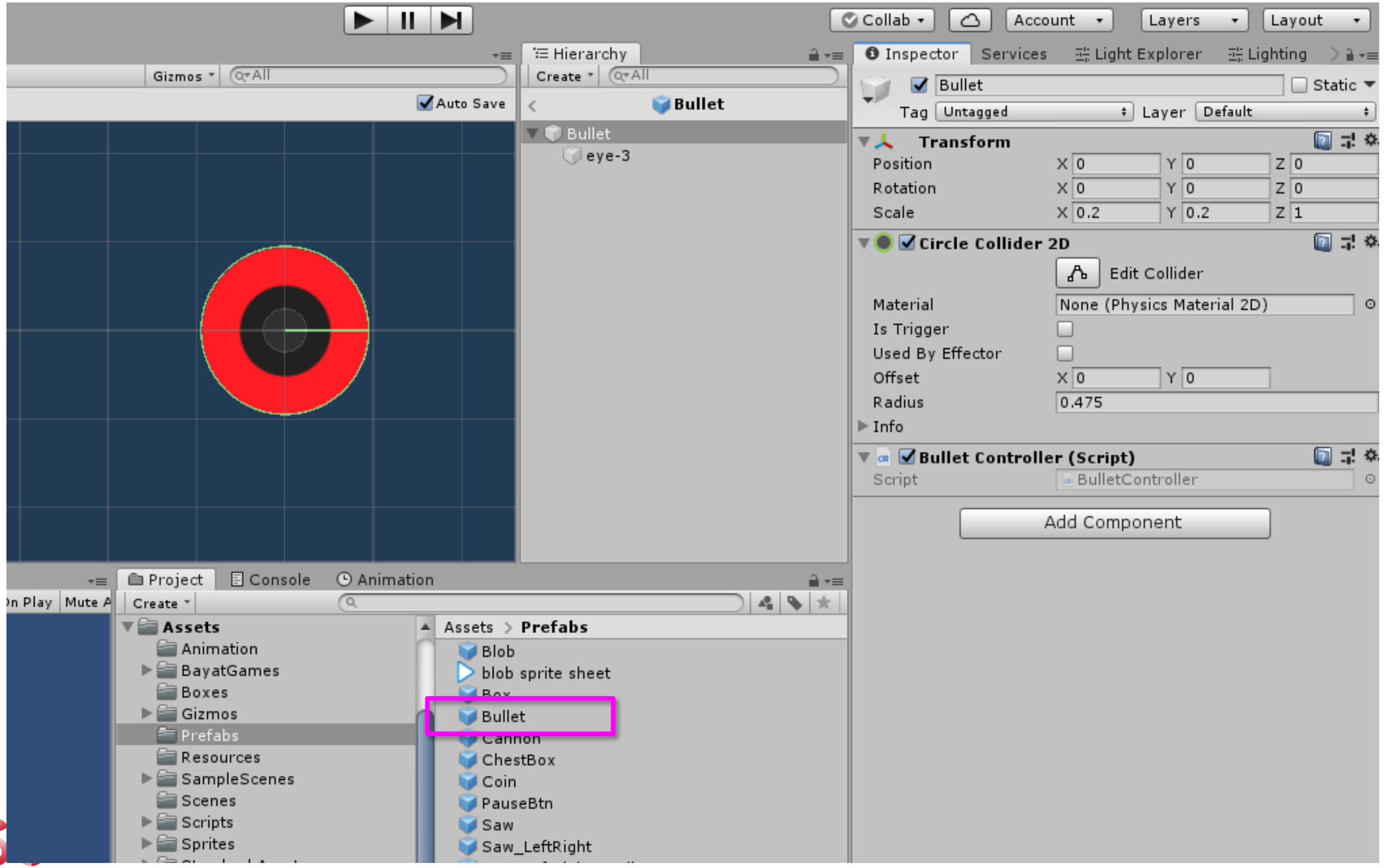
Dangerous Kave 12

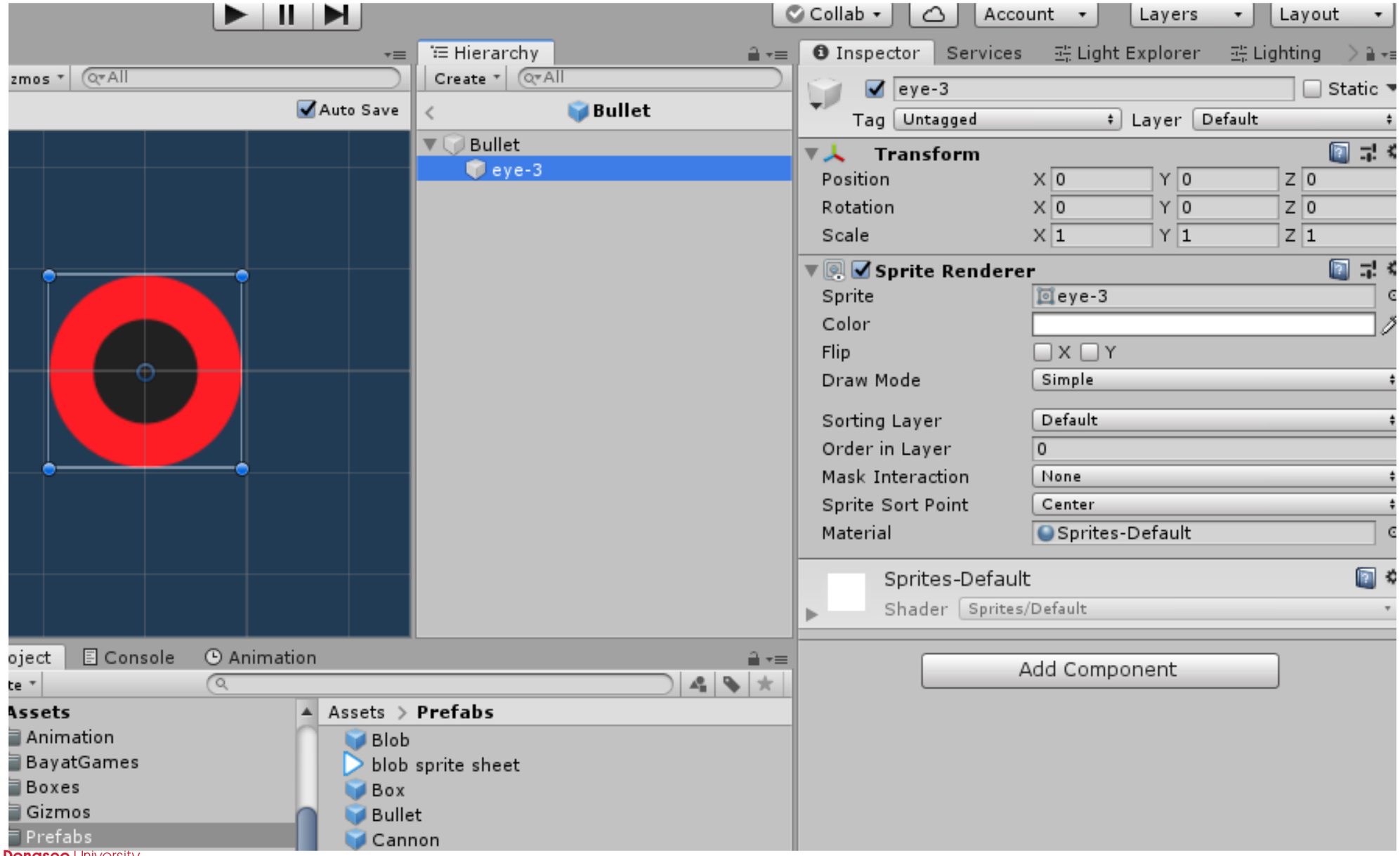
Cannon fires a bullet

jintaeks@dongseo.ac.kr

May, 2020

DSU Dongseo University 동서대학교


Collab
Account
Layers
Layout
Hierarchy
Gizmos
All
Auto Save
Create
Bullet
eye-3
Inspector
Services
Light Explorer
Lighting
Static
Tag
Untagged
Layer
Default
Transform
Position
Rotation
Scale
Circle Collider 2D
Edit Collider
Material
None (Physics Material 2D)
Is Trigger
Used By Effector
Offset
Radius
0.475
Info
Bullet Controller (Script)
Script
BulletController
Add Component
Project
Console
Animation
Assets
Animation
BayatGames
Boxes
Gizmos
Prefabs
Resources
SampleScenes
Scenes
Scripts
Sprites
Assets > Prefabs
Blob
blob sprite sheet
Bullet
Cannon
ChestBox
Coin
PauseBtn
Saw
Saw_LeftRight


Collab
Account
Layers
Layout
Hierarchy
Create
All
Bullet
eye-3
Auto Save
Inspector
Services
Light Explorer
Lighting
eye-3
Static
Tag
Untagged
Layer
Default
Transform
Position
Rotation
Scale
Sprite Renderer
Sprite
Color
Flip
Draw Mode
Simple
Sorting Layer
Default
Order in Layer
Mask Interaction
None
Sprite Sort Point
Center
Material
Sprites-Default
Shader
Sprites/Default
Add Component
Console
Animation
Assets
Animation
BayatGames
Boxes
Gizmos
Prefabs
Assets > Prefabs
Blob
blob sprite sheet
Box
Bullet
Cannon

BoxBehaviour

```
private Vector2 _instantaneousVelocity = Vector2.zero;
    public Vector2 velocity
    {
        get { return _instantaneousVelocity; }
    }

    public Vector2 maxVelocity
    {
        get { return _maxVelocity; }
    }

    private GameObject _player; // reference to the player character

    [SerializeField]
    private BoxCollider2D _boxCollider;
```

```
 void _Update_StateIDLE()
    {
        // Retrieve all colliders we have intersected after velocity has been applied.
        RaycastHit2D[] hits = Physics2D.BoxCastAll(transform.position, _boxCollider.size, 0,
new Vector2(0, 0));
        if (hits.Length >= 1)
        {
            foreach (RaycastHit2D hit in hits)
            {
                // Ignore our own collider.
                if (hit.transform == transform)
                    continue;

                if (hit.transform.gameObject.IsMovingObject())
                {
                    bool isStopIterate = /*virtual*/OnStateIdle_Hit(hit);
                    if( isStopIterate )
                        break;
                }
            }//foreach
        }
        _UpdateCollision();
    }
```

```
void _UpdateCollision()
    {
        /*virtual*/ OnPreCollision();

        if (_movingState == EState.MOVING)
        {
            _curVelocity = Vector2.MoveTowards(_curVelocity, _maxVelocity, _acceleration * Time.deltaTime);
            transform.Translate(_curVelocity * Time.deltaTime);
        }

        _isGrounded = false;
        bool isInAir = true;

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

        _numCollision = hits.Length;

        foreach (Collider2D hit in hits)
        {
            // Ignore our own collider.
            if (hit.transform == transform)
                continue;

            //if( hit.gameObject.IsMovingObject())
            //{
            //    _stateTimer = 0.0f; // initialize timer when there is a collision with 'Player' or 'Box'
            //}
```

```
            isInAir = false;
            ColliderDistance2D colliderDistance = hit.Distance(_boxCollider);

            if (colliderDistance.isOverlapped)
            {
                /*virtual*/ OnOverlapped( hit, colliderDistance);

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

        if (isInAir != _isInAir)
        {
            _isInAir = isInAir;
        }

        /*virtual*/ OnPostCollision();
    }

    void _StateMOVING_UpdateCollision()
    {
        _UpdateCollision();
    }
```

BoxController

```
35
36      override public void OnPreCollision()
37      {
38          if (base.movingState == EState.MOVING )
39              isContactSawTemp = false;
40      }
41
42      override public void OnOverlapped(Collider2D hit, ColliderDistance2D colliderDistance)
43      {
44          if (hit.transform.CompareTag("Saw"))
45          {
46              isContactSawTemp = true;
47              _grindContactPos = colliderDistance.pointB;
48          }
49
50          if (hit.transform.CompareTag("Player"))
51          {
52              hit.transform.Translate(colliderDistance.pointB - colliderDistance.pointA);
53          }
54          else
```

BulletController

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BulletController : MonoBehaviour
{
    private Vector2 _velocity;
    public Vector2 velocity
    {
        get { return _velocity; }
        set { _velocity = value; }
    }
    private CircleCollider2D _circleCollider;
    private float _lifeTimer = 0.0f;
    private Camera _mainCamera = null;

    // Start is called before the first frame update
    void Start()
    {
        _circleCollider = GetComponent<CircleCollider2D>();
        _mainCamera = Camera.main;
    }
```

BulletController

```
    // Update is called once per frame
    void Update()
    {
        _lifeTimer += Time.deltaTime;
        Vector3 screenPoint = _mainCamera.WorldToViewportPoint(transform.position);
        bool isInViewport = (screenPoint.z > 0 && screenPoint.x > 0 && screenPoint.x < 1 && screenPoint.y >
0 && screenPoint.y < 1);
        if (isInViewport == false)
        {
            Destroy(gameObject);
        }

        transform.Translate(_velocity * Time.deltaTime);
        Collider2D[] hits = Physics2D.OverlapCircleAll(transform.position
            , _circleCollider.radius*transform.localScale.x);
        foreach (Collider2D hit in hits)
        {
            // Ignore our own collider.
            if (hit.transform == transform)
                continue;
            if (hit.transform.CompareTag("Player"))
            {
                // @TODO: must be moved into CharacterController2D
                // @date: 20200428
                Destroy(hit.gameObject);
            }
```

BulletController

```
foreach (Collider2D hit in hits)
        {
            // Ignore our own collider.
            if (hit.transform == transform)
                continue;
            if (hit.transform.CompareTag("Player"))
            {
                // @TODO: must be moved into CharacterController2D
                // @date: 20200428
                Destroy(hit.gameObject);
            }
            LevelManager.CreateEffect(LevelManager.EffectType.SmallImpact, transform.position,
transform.rotation);
            Destroy(gameObject);
            break;
        }
    }
}
```

SawController

```
11
12      override public void OnOverlapped(Collider2D hit, ColliderDistance2D colliderDistance)
13      {
14          if (hit.gameObject.IsMovingObject())
15          {
16              hit.transform.Translate(colliderDistance.pointB - colliderDistance.pointA);
17          }
18          else
19          {
20              if( base.movingState == EState.MOVING )
21                  transform.Translate(colliderDistance.pointA - colliderDistance.pointB);
22          }
23      }
24  }
```

CannonController

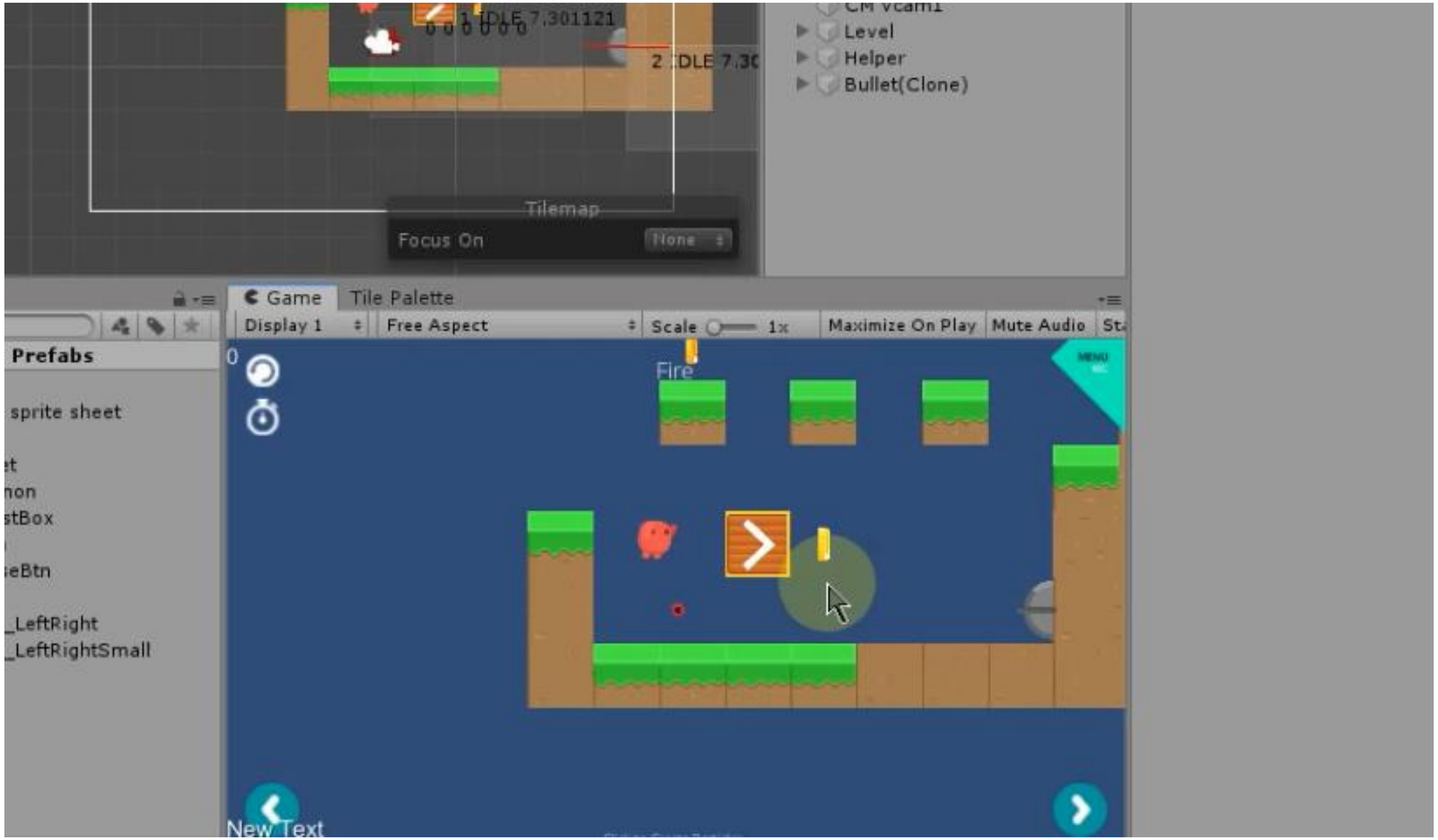
```
public class CannonController : BoxBehaviour
{
    public Transform _muzzlePoint;
    public Transform _bullet;
    public float _fireInterval = 2.0f; // unit: second
    private float _fireTimer = 0.0f;

    // Start is called before the first frame update
    override public void OnStart()
    {
        if (_bullet == null)
            _bullet = Resources.Load("Bullet") as Transform;
    }
```

```
 // Update is called once per frame
    override public void OnUpdate(EState movingState, float stateTimer)
    {
        // need to delete: jintaeks on 20200428
        //if (Input.GetKeyDown(KeyCode.Alpha1))
        //{
        //    if (_bullet)
        //    {
        //        Transform b = Instantiate(_bullet, _muzzlePoint.position,
Quaternion.identity) as Transform;
        //        BulletController bc = b.GetComponent<BulletController>();
        //        bc.velocity = maxVelocity;
        //    }
        //}
        _fireTimer += Time.deltaTime;
        if (_fireTimer >= _fireInterval)
        {
            _fireTimer = 0.0f;
            _FireBullet();
        }
    }
```

```
    override public void OnOverlapped(Collider2D hit, ColliderDistance2D colliderDistance)
    {
        if (hit.transform.CompareTag("Player"))
        {
            hit.transform.Translate(colliderDistance.pointB - colliderDistance.pointA);
        }
        else
        {
            if( base.movingState == EState.MOVING )
                transform.Translate(colliderDistance.pointA - colliderDistance.pointB);
        }
    }

    private void _FireBullet()
    {
        Transform b = Instantiate(_bullet, _muzzlePoint.position, Quaternion.identity) as
Transform;
        BulletController bc = b.GetComponent<BulletController>();
        bc.velocity = maxVelocity;
    }
}
```



DSU Dongseo University 동서대학교

MY **BRIGHT** FUTURE

DSU Dongseo University 동서대학교

DSU Dongseo University 동서대학교