Eagle Script Code Sample

Deven Smith

```
using UnityEngine;
using System.Collections;

public class EagleScript : MonoBehaviour {
	Ages age;
	public float moveSpeed;
	public float eggMoveSpeed;
	public float normalMoveSpeed;
	public float oldMoveSpeed;

	public GameObject egg;
	public GameObject chick;
	public GameObject adult;
	public GameObject great;

	public int startingAge = -1;

	public GameObject[] flightPath;
	int curFPPoint;
	public GameObject target;
	public float rotateSpeed = 3;
	// Use this for initialization

	public BoxCollider myCollider;
	void Start () {

		if(startingAge > 0 && startingAge < 4)
			for(int i = 0; i < startingAge; i++)
				age++;
		else
			age = Ages.baby;


		curFPPoint = 0;
		target = flightPath[0];
		Aged(0);
	}

	// Update is called once per frame
	void Update () {
		if(age == Ages.normal || age == Ages.old)
		{
			//transform.LookAt(target.transform.position);
			float step = rotateSpeed * Time.deltaTime;
			Vector3 targetRotation = (target.transform.position - transform.position).normalized;
			Quaternion lookRotation = Quaternion.LookRotation(targetRotation);
			transform.rotation = Quaternion.RotateTowards(transform.rotation, lookRotation, step);
			//transform.Translate(transform.forward*moveSpeed*Time.deltaTime);
			transform.position = transform.position + transform.forward*moveSpeed*Time.deltaTime;

			//goes from first point in flight path to next and goes from last fp point to first
			if(Vector3.Distance(target.transform.position, transform.position) < 3)
			{
				curFPPoint++;
				if(curFPPoint >= flightPath.Length)
					curFPPoint = 0;


				target = flightPath[curFPPoint];
```

```
                }
            }
        }

        void Aged(int effect)
        {
            if(effect > 0)
            {
                if(age == Ages.old)
                {
                    //Destroy (this.gameObject);
                }
                else
                    age++;
            }
            else if (effect < 0)
            {
                if(age == Ages.baby)
                {
                    //Destroy (this.gameObject);
                }
                else
                    age--;
            }

            switch(age)
            {
            case Ages.baby:
                transform.tag = "Egg";
                moveSpeed = 0;
                egg.SetActive(true);
                chick.SetActive(false);
                adult.SetActive(false);
                great.SetActive(false);
                myCollider.size = new Vector3(.71f,.57f,.67f);
                break;
            case Ages.young:
                transform.tag = "Eagle";
                moveSpeed = 0.0f;
                egg.SetActive(false);
                chick.SetActive(true);
                adult.SetActive(false);
                great.SetActive(false);
                myCollider.size = new Vector3(.71f,.57f,.67f);
                break;
            case Ages.normal:
                transform.tag = "Eagle";
                moveSpeed = 10.0f;
                egg.SetActive(false);
                chick.SetActive(false);
                adult.SetActive(true);
                great.SetActive(false);
                myCollider.size = new Vector3(1.9f,.57f, 1.61f);
                break;
            case Ages.old:
                transform.tag = "Eagle";
                moveSpeed = 5.0f;
                egg.SetActive(false);
                chick.SetActive(false);
                adult.SetActive(false);
                great.SetActive(true);
                myCollider.size = new Vector3(6,.57f,3.37f);
```

```
                        break;
                default:
                        break;
                }
        }
}
```

Jet Flight Script Code Sample

Deven Smith

```
using UnityEngine;
using System.Collections;

public class JetFlight : MonoBehaviour {

        public KeyCode accelerateKey;
        public KeyCode decelerateKey;
        public KeyCode rollLeftKey;
        public KeyCode rollRightKey;
        public KeyCode yawLeftKey;
        public KeyCode yawRightKey;
        public KeyCode pitchUpKey;
        public KeyCode pitchDownKey;

        public float minimumVelocity;
        public float maxVelocity;
        public float drag = 1;

        float acceleration = 0.0f;
        float maxAcceleration;
        public float velocity = 0.0f;
        public GUIText speedText;
        public Vector3 rotation;

        //Vector3 direction;

        // Use this for initialization
        void Start () {
                maxAcceleration = maxVelocity;
        }

        // Update is called once per frame
        void Update () {

                //Camera.main.transform.position = new Vector3(0, 0.6604233f, -0.6326837f);

                //set a vector3 to handle the rotation then adjust the transform to match
                velocity = rigidbody.velocity.magnitude;

                //direction = transform.forward;

                //speed up or down
                if(Input.GetKey(accelerateKey))
                {
                        if(acceleration < maxAcceleration)
                                acceleration += .2f;
                }
                if(Input.GetKey(decelerateKey))
                {
                        acceleration -= .1f;
                        if(acceleration < 0)
                                acceleration = 0;
                }

                if(acceleration > 0)
                {
                        rigidbody.AddForce(transform.forward * acceleration, ForceMode.Force);
                        //acceleration -= .1f;
                }
```

```
				//roll left or right
				if(Input.GetKey(rollLeftKey))
				{
					rotation = new Vector3(rotation.x, rotation.y, rotation.z + 1);
				}
				if(Input.GetKey(rollRightKey))
				{
					rotation = new Vector3(rotation.x, rotation.y, rotation.z - 1);
				}

				//yaw/turn left or right
				if(Input.GetKey(yawLeftKey))
				{
					rotation = new Vector3(rotation.x, rotation.y - .5f, rotation.z);
				}
				if(Input.GetKey(yawRightKey))
				{
					rotation = new Vector3(rotation.x, rotation.y + .5f, rotation.z);
				}

				//pitch up or down
				if(Input.GetKey(pitchUpKey))
				{
					rotation = new Vector3(rotation.x - .5f, rotation.y , rotation.z);
				}
				if(Input.GetKey(pitchDownKey))
				{
					rotation = new Vector3(rotation.x + .5f, rotation.y , rotation.z);
				}


				//keep the velocity in bounds
				if(rigidbody.velocity.magnitude < minimumVelocity)
				{
					rigidbody.drag = 0;
				}
				else
				{
					rigidbody.drag = drag;
				}

				if(rigidbody.velocity.magnitude > maxVelocity)
				{

				}
				transform.eulerAngles = rotation;
				rigidbody.velocity = transform.forward * velocity;
				speedText.text = rigidbody.velocity.magnitude + " MPH";

		}
}
```

Rag Doll Script Code Sample

Deven Smith

```
using UnityEngine;
using System.Collections;

public class RagDollScript : MonoBehaviour {

	GameObject hitByBullet;
	public int impactMultiplier = 1000;

	/*// Use this for initialization
	void Start () {

	}

	// Update is called once per frame
	void Update () {

	}*/

	void SetImpact(GameObject bullet)
	{
		hitByBullet = bullet;
		BulletScript bs = hitByBullet.GetComponent<BulletScript>();

		//GameObject child = (GameObject) Instantiate(ragdoll, transform.position, transform.rotation);
		//child.transform.parent = transform;

		//child.SetActive(true);
		GameObject root;
		/*if(typeOfAttack == AttackTypes.Straight)
			{
				foreach(Transform c in transform)
				{
					if(c.name == "ranger_Hips")
					{
						Debug.Log("found hips");
						root = c.gameObject;
						root.rigidbody.AddForce(-transform.forward * 10);
					}
				}*/
		root = transform.FindChild("ranger_Reference/ranger_Hips").gameObject;
		//root.rigidbody.AddForce(hitByBullet.transform.forward * 1000,ForceMode.Acceleration);


		switch(bs.attackType)
		{
		case AttackTypes.Straight :
			root.rigidbody.AddForce(/*hitByBullet.transform.forward* */hitByBullet.transform.forward *
impactMultiplier,ForceMode.Acceleration);
			break;
		case AttackTypes.Horizontal :
			root.rigidbody.AddForce(/*hitByBullet.transform.forward* */hitByBullet.transform.right *
impactMultiplier,ForceMode.Acceleration);
			break;
		case AttackTypes.Verticle :
			root.rigidbody.AddForce(/*hitByBullet.transform.forward* */hitByBullet.transform.up *
impactMultiplier,ForceMode.Acceleration);

			break;
		default:
```

```
                        root.rigidbody.AddForce(/*hitByBullet.transform.forward* */hitByBullet.transform.forward *
impactMultiplier,ForceMode.Acceleration);
                        break;
                }

                //Debug.Log(hitByBullet.rigidbody.velocity);
                //root.rigidbody.AddForceAtPosition(hitByBullet.transform.forward, hitByBullet.transform.position,
ForceMode.Acceleration);
                Destroy(hitByBullet);
        }

        void SetImpact(Vector3 Direction)
        {
                GameObject root;
                root = transform.FindChild("ranger_Reference/ranger_Hips").gameObject;
                root.rigidbody.AddForce(/*hitByBullet.transform.forward* */Direction * impactMultiplier,ForceMode.Acceleration);
        }
}
```

Tile Script Code Sample

Deven Smith

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

public class TileScript : MonoBehaviour {

        GameObject monster;

        public int militaryHealth = 1;
        public Vector2 mHealthRange;

        public int militaryArmour = 0;
        public Vector2 mArmourRange;
        public int militaryResistance = 0;
        public Vector2 mResistanceRange;
        public int militaryInsulation = 0;
        public Vector2 mInsulationRange;

        public int militaryPhysical = 0;
        public Vector2 mPhysicalRange;
        public int militaryFire = 0;
        public Vector2 mFireRange;
        public int militaryElectric = 0;
        public Vector2 mElectricRange;

        public int pointValue = 0;
        public Vector2 pointValueRange;

        public bool destroyed = false;

        public GameObject[] tiles;
        public List<GameObject> Neighbors;



        // Use this for initialization
        void Start ()
        {
                //Destroy(transform.GetChild(0).gameObject);
                SetTiles ();
```

```
                    //SetMilitary();
                    pointValue = (int) Random.Range(pointValueRange.x, pointValueRange.y);

                    FindNeighbors();

          }

          // Update is called once per frame
          void Update () {
                    if(militaryHealth <= 0)
                    {
                              if(destroyed == false)
                              {
                                        //change to the destroyed tile texture
                                        if(transform.GetChild(0))
                                                  Destroy(transform.GetChild(0).gameObject);

                                        GameObject tile = (GameObject) Instantiate(tiles[3], transform.position,
tiles[3].transform.rotation);
                                        tile.transform.parent = transform;
                                        destroyed = true;
                              }
                    }
          }

          void OnMouseDown()
          {
                    //Debug.Log("clicked a tile");
                    GameObject monster = GameObject.FindGameObjectWithTag("Monster");
                    MonsterMoverScript monsterMover = monster.GetComponent<MonsterMoverScript>();
                    monsterMover.SendMessage("SetTarget", gameObject);

          }

          void FindNeighbors()
          {
                    //raycast out of the six sides and if a tile is returned then add it as a neighbor

                    Quaternion originalFacing = transform.rotation;
                    RaycastHit hit;
                    float rayRange = 10.0f;

                    if(Physics.Raycast(transform.position, transform.forward, out hit, rayRange))
                    {
                              if(hit.collider.tag == "Tile")
                                        Neighbors.Add(hit.collider.transform.gameObject);
                              else
                              {
                                        Debug.DrawLine(transform.position, transform.forward*rayRange, Color.green, 100);
                              }
                    }

                    transform.eulerAngles = transform.eulerAngles + new Vector3(0,60,0);
                    //transform.rotation = new Quaternion(transform.rotation.x, transform.rotation.y + 60, transform.rotation.z,
transform.rotation.w);
                    if(Physics.Raycast(transform.position, transform.forward, out hit, rayRange))
                    {
                              if(hit.collider.tag == "Tile")
                                        Neighbors.Add(hit.collider.transform.gameObject);
                              else
                              {
                                        Debug.DrawLine(transform.position, transform.forward*rayRange, Color.green, 100);
```

```
			}
		}

		transform.eulerAngles = transform.eulerAngles + new Vector3(0,60,0);
		//transform.rotation = new Quaternion(transform.rotation.x, transform.rotation.y + 60, transform.rotation.z,
transform.rotation.w);
		if(Physics.Raycast(transform.position, transform.forward, out hit, rayRange))
		{
			if(hit.collider.tag == "Tile")
				Neighbors.Add(hit.collider.transform.gameObject);
			else
			{
				Debug.DrawLine(transform.position, transform.forward*rayRange, Color.green, 100);
			}
		}

		transform.eulerAngles = transform.eulerAngles + new Vector3(0,60,0);
		//transform.rotation = new Quaternion(transform.rotation.x, transform.rotation.y + 60, transform.rotation.z,
transform.rotation.w);
		if(Physics.Raycast(transform.position, transform.forward, out hit, rayRange))
		{
			if(hit.collider.tag == "Tile")
				Neighbors.Add(hit.collider.transform.gameObject);
			else
			{
				Debug.DrawLine(transform.position, transform.forward*rayRange, Color.green, 100);
			}
		}

		transform.eulerAngles = transform.eulerAngles + new Vector3(0,60,0);
		//transform.rotation = new Quaternion(transform.rotation.x, transform.rotation.y + 60, transform.rotation.z,
transform.rotation.w);
		if(Physics.Raycast(transform.position, transform.forward, out hit, rayRange))
		{
			if(hit.collider.tag == "Tile")
				Neighbors.Add(hit.collider.transform.gameObject);
			else
			{
				Debug.DrawLine(transform.position, transform.forward*rayRange, Color.green, 100);
			}
		}

		transform.eulerAngles = transform.eulerAngles + new Vector3(0,60,0);
		//transform.rotation = new Quaternion(transform.rotation.x, transform.rotation.y + 60, transform.rotation.z,
transform.rotation.w);
		if(Physics.Raycast(transform.position, transform.forward, out hit, rayRange))
		{
			if(hit.collider.tag == "Tile")
				Neighbors.Add(hit.collider.transform.gameObject);
			else
			{
				Debug.DrawLine(transform.position, transform.forward*rayRange, Color.green, 100);
			}
		}
		transform.rotation = originalFacing;
	}


	void SetTiles()
	{
		int randomNumber = Random.Range(0, tiles.Length-1);
```

```
			GameObject tile = (GameObject) Instantiate(tiles[randomNumber], transform.position, 
tiles[randomNumber].transform.rotation);

			if(randomNumber == 1)//barracks
				SetMilitary(new Vector2(100,1000), new Vector2(10,50), new Vector2(10,50), new Vector2(10,50), new 
Vector2(10,100), new Vector2(10,100), new Vector2(10,100));
			if(randomNumber == 2)//buildings
				SetMilitary(new Vector2(100,500), new Vector2(10,20), new Vector2(10,20), new Vector2(10,20), new 
Vector2(10,20), new Vector2(10,20), new Vector2(10,20));

			tile.transform.parent = transform;
		}

		void SetMilitary(Vector2 health, Vector2 armour, Vector2 resistance, Vector2 insulation, Vector2 physical, Vector2 fire, Vector2 
electric)
		{
			//set up the military stats
			mHealthRange = health;
			mArmourRange = armour;
			mResistanceRange = resistance;
			mInsulationRange = insulation;
			mPhysicalRange = physical;
			mFireRange = fire;
			mElectricRange = electric;

			militaryHealth += (int) Random.Range(mHealthRange.x, mHealthRange.y);

			//defenses
			militaryArmour = (int) Random.Range(mArmourRange.x, mArmourRange.y);
			militaryResistance = (int) Random.Range(mResistanceRange.x, mResistanceRange.y);
			militaryInsulation = (int) Random.Range(mInsulationRange.x, mInsulationRange.y);

			//offenses
			militaryPhysical = (int) Random.Range(mPhysicalRange.x, mPhysicalRange.y);
			militaryFire = (int) Random.Range(mFireRange.x, mFireRange.y);
			militaryElectric = (int) Random.Range(mElectricRange.x, mElectricRange.y);
		}
}
```

Lightning Script Code Sample

Steven Hoover

```
using UnityEngine;
using System.Collections;

public enum LightningState {Waiting, FirstStrike, SecondStrike, ThirdStrike, ComeDown};

public class LightningScript : MonoBehaviour {

		public Light first;
		public Light second;
		LightningState state;
		//times
		public float timeUntillLightning= 10;
		public float timeBetweenStrikes1and2 = 0.2f;
		public float timeBetweenStrikes2and3 = 0.2f;
		public float timeLightningStrikeOneDur = 0.3f;
		public float timeLightningStrikeTwoDur = 0.3f;
		public float timeLightningStrikeThreeDur = 0.3f;
		float timeSinceLastStrike =0;

		public float interval = 0.1f;
public float firstStrikeBrightnessMax = 0.4f;
public float secondStrikeBrightnessMax = 0.5f;
public float thirdStrikeBrightnessMax = 1.2f;
public bool striking = false;
// Use this for initialization
void Start () {

}

// Update is called once per frame
void Update () {
        timeSinceLastStrike += Time.deltaTime;

        switch (state)
        {
        case LightningState.Waiting:
                if(timeSinceLastStrike > timeUntilLightning)
                {
                        striking = true;
                        timeSinceLastStrike = 0;
                        state = LightningState.FirstStrike;
                }
                break;

        case LightningState.FirstStrike:
                if(first.intensity < firstStrikeBrightnessMax)
                        first.intensity += interval;
                if(timeSinceLastStrike > timeLightningStrikeOneDur)
                {
                        timeSinceLastStrike = 0;
                        state = LightningState.SecondStrike;
                }
                break;

        case LightningState.SecondStrike:
                if(first.intensity < interval && timeSinceLastStrike > timeBetweenStrikes1and2)
                {
                        if( second.intensity < secondStrikeBrightnessMax) second.intensity += interval;
                        if( timeSinceLastStrike > timeBetweenStrikes1and2 + timeLightningStrikeTwoDur)
                        {
                                timeSinceLastStrike = 0;
                                state = LightningState.ThirdStrike;
                        }
                }
                else
                {
                        first.intensity -= interval;
                }
                break;

        case LightningState.ThirdStrike:
                if(second.intensity < interval && timeSinceLastStrike > timeBetweenStrikes2and3)
                {
                        if( first.intensity < thirdStrikeBrightnessMax) first.intensity += interval;
                        if( timeSinceLastStrike > timeLightningStrikeThreeDur)
                        {
                                state = LightningState.ComeDown;
                                timeSinceLastStrike = 0;
                        }
                }
                else
                {
                    second.intensity -= interval;
                }
                break;

            case LightningState.ComeDown:
                if(first.intensity > 0)
                {
                    first.intensity -= interval;
                }
                else if(second.intensity > 0)
                {
                    second.intensity -= interval;
                }
                else
                {
                    striking = false;
                    timeSinceLastStrike = 0;
                    state = LightningState.Waiting;
                }
                break;

        }
    }
}
```

Networking Manager Script Code Sample

Steven Hoover

```
using UnityEngine;
using System.Collections;

public class NetworkingManagerScript : MonoBehaviour {

    string gameName = "Tag";

    bool refreshing = false;
    HostData [] hostData;

    float bX;
    float bY;
    float bW;
    float bH;

    public GameObject playerPrefab;
    public Transform spawnObject;

    // Use this for initialization
    void Start () {
        bX = Screen.width * 0.05f;
        bY = Screen.width * 0.05f;
        bW = Screen.width * 0.1f;
        bH = Screen.width * 0.1f;
    }

    // Update is called once per frame
    void Update () {
        if(refreshing)
        {
            if(MasterServer.PollHostList().Length > 0)
            {
                refreshing = false;
                hostData = MasterServer.PollHostList();
```

```
                    }
                }
            }

            void SpawnPlayer()
            {
                //Network.Instantiate(playerPrefab, spawnObject.position, Quaternion.identity, 0);
            }

            void StartServer()
            {
                print("starting");
                Network.InitializeServer(32, 25001, !Network.HavePublicAddress());
                MasterServer.RegisterHost(gameName, "Tag", "HI, DONT");
            }

            void OnServerInitialized()
            {
                Application.LoadLevel(1);
            }

            void OnConnectedToServer()
            {
                Application.LoadLevel(1);
            }

            void OnMasterServerEvent(MasterServerEvent mse)
            {
                if(mse == MasterServerEvent.RegistrationSucceeded)
                {
                    Debug.Log("win");
                }
            }

            void RefreshHostList()
            {
                MasterServer.RequestHostList(gameName);
                refreshing = true;
            }
            //GUI
            void OnGUI()
            {
                if(!Network.isClient && !Network.isServer)
                {
                    if( GUI.Button(new Rect(bX, bY, bW, bH), "Start Server") )
                    {
                        StartServer();
                    }

                    if( GUI.Button(new Rect(bX, bY * 1.2f + bH, bW, bH), "Refresh Host") )
                    {
                        RefreshHostList();
                    }
                    if(hostData != null)
                    {
                        for( int i =0; i<hostData.Length; i++)
                        {
                            if(GUI.Button (new Rect(bX * 1.5f + bW, bY * 1.2f + (bH * i), bW*3, bH*0.5f),
hostData[i].gameName) )
                            {
                                Network.Connect (hostData[i]);
                            }
                }
            }
        }
    }
}
```

Player Control 2D Script Code Sample

Steven Hoover

```
using UnityEngine;
using System.Collections;

public class PlayerControl2D : MonoBehaviour
{
    [HideInInspector]
    public bool facingRight = true;            // For determining which way the player is currently facing.
    [HideInInspector]
    public bool jump = false;                      // Condition for whether the player should jump.


    public float moveForce = 365f;             // Amount of force added to move the player left and right.
    public float maxSpeed = 5f;                    // The fastest the player can travel in the x axis.
    public AudioClip[] jumpClips;              // Array of clips for when the player jumps.
    public float jumpForce = 1000f;            // Amount of force added when the player jumps.
    public bool slam = false;
    public Camera mainCam;
//  public AudioClip[] taunts;                     // Array of clips for when the player taunts.
//  public float tauntProbability = 50f;       // Chance of a taunt happening.
//  public float tauntDelay = 1f;              // Delay for when the taunt should happen.
//
//
//  private int tauntIndex;                        // The index of the taunts array indicating the most recent taunt.
    private Transform groundCheck;             // A position marking where to check if the player is grounded.
    private bool grounded = false;             // Whether or not the player is grounded.
//  private Animator anim;                             // Reference to the player's animator component.


    void Awake()
    {
        // Setting up references.
        groundCheck = transform.Find("groundCheck");
        //anim = GetComponent<Animator>();
    }


    void Update()
    {
        // The player is grounded if a linecast to the groundcheck position hits anything on the ground layer.
        grounded = Physics2D.Linecast(transform.position, groundCheck.position, 1 << LayerMask.NameToLayer("Ground"));

        if( slam && grounded )
        {
            mainCam.GetComponent<CameraShake>().Shake();
            slam = false;
        }

        // If the jump button is pressed and the player is grounded then the player should jump.
        if(Input.GetButtonDown("Jump") && grounded)
            jump = true;
        if( Input.GetKey( KeyCode.E ) && !grounded)
        {
            Slam();
```

```
		}
	}

	void Slam()
	{
		slam = true;
		rigidbody2D.AddForce( new Vector2(0,-150) );
		//mainCam.GetComponent<CameraShake>().Shake();
	}

	void UpdateSlam()
	{

	}
	void FixedUpdate ()
	{
		// Cache the horizontal input.
		float h = Input.GetAxis("Horizontal");
		h *= -1;
		// The Speed animator parameter is set to the absolute value of the horizontal input.
		//anim.SetFloat("Speed", Mathf.Abs(h));

		// If the player is changing direction (h has a different sign to velocity.x) or hasn't reached maxSpeed yet...
		if(h * rigidbody2D.velocity.x < maxSpeed)
			// ... add a force to the player.
			rigidbody2D.AddForce(Vector2.right * h * moveForce);

		// If the player's horizontal velocity is greater than the maxSpeed...
		if(Mathf.Abs(rigidbody2D.velocity.x) > maxSpeed)
			// ... set the player's velocity to the maxSpeed in the x axis.
			rigidbody2D.velocity = new Vector2(Mathf.Sign(rigidbody2D.velocity.x) * maxSpeed, rigidbody2D.velocity.y);

		// If the input is moving the player right and the player is facing left...
		if(h > 0 && !facingRight)
			// ... flip the player.
			Flip();
		// Otherwise if the input is moving the player left and the player is facing right...
		else if(h < 0 && facingRight)
			// ... flip the player.
			Flip();

		// If the player should jump...
		if(jump)
		{
			// Set the Jump animator trigger parameter.
			//anim.SetTrigger("Jump");

			// Play a random jump audio clip.
			int i = Random.Range(0, jumpClips.Length);
			//AudioSource.PlayClipAtPoint(jumpClips[i], transform.position);

			// Add a vertical force to the player.
			rigidbody2D.AddForce(new Vector2(0f, jumpForce));

			// Make sure the player can't jump again until the jump conditions from Update are satisfied.
			jump = false;
		}
	}


	void Flip ()
	{
					// Switch the way the player is labelled as facing.
					facingRight = !facingRight;

					// Multiply the player's x local scale by -1.
					Vector3 theScale = transform.localScale;
					theScale.x *= -1;
					transform.localScale = theScale;
			}


//		public IEnumerator Taunt()
//		{
//				// Check the random chance of taunting.
//				float tauntChance = Random.Range(0f, 100f);
//				if(tauntChance > tauntProbability)
//				{
//						// Wait for tauntDelay number of seconds.
//						yield return new WaitForSeconds(tauntDelay);
//
//						// If there is no clip currently playing.
//						if(!audio.isPlaying)
//						{
//								// Choose a random, but different taunt.
//								tauntIndex = TauntRandom();
//
//								// Play the new taunt.
//								audio.clip = taunts[tauntIndex];
//								audio.Play();
//						}
//				}
//		}
//
//
//		int TauntRandom()
//		{
//				// Choose a random index of the taunts array.
//				int i = Random.Range(0, taunts.Length);
//
//				// If it's the same as the previous taunt...
//				if(i == tauntIndex)
//						// ... try another random taunt.
//						return TauntRandom();
//				else
//						// Otherwise return this index.
//						return i;
//		}
}
```

Space Script Code Sample

Steven Hoover

```
using UnityEngine;
using System.Collections;

public enum SpaceType {Ship, Trash, Starting};

public class SpaceScript : MonoBehaviour {

		public GameFlow game;

		public SpaceType spaceType;
public bool used;
public int pieceIDOnMe;

bool adjacentsFound = false;
public SpaceScript top, left, bottom, right;
public bool haveFault = false;
public int exposedCount =0;
// Use this for initialization
void Start () {

}

// Update is called once per frame
void Update ()
{
        if(!adjacentsFound)
        {
                FindAdjacents();
        }


        if(used)
        {
                if(spaceType == SpaceType.Trash)
                {
                        game.DeletePiece(pieceIDOnMe);
                        used = false;
                }
                //check for faults
                haveFault = false;
                FindFault(top, 2, 0);
                FindFault(left, 1, 3);
                FindFault(bottom, 0, 2);
                FindFault(right, 3 , 1);
        }
        else
        {
                haveFault = false;
                exposedCount =0;
        }
}

void FindAdjacents()
{
        RaycastHit hit;
        //top
        if(Physics.Raycast(transform.position,transform.up,out hit,1.5f) )
        {
                if(hit.collider.tag == "space")
                {
                        top = hit.collider.gameObject.GetComponent<SpaceScript>();
                }
        }

        //left
        if(Physics.Raycast(transform.position,-transform.right,out hit,1.5f) )
        {
                if(hit.collider.tag == "space")
                {
                        left = hit.collider.gameObject.GetComponent<SpaceScript>();
                }
        }
```

```
			//bottom
			if(Physics.Raycast(transform.position,-transform.up,out hit,1.5f) )
			{
				if(hit.collider.tag == "space")
				{
					bottom = hit.collider.gameObject.GetComponent<SpaceScript>();
				}
			}

			//right
			if(Physics.Raycast(transform.position,transform.right,out hit,1.5f) )
			{
				if(hit.collider.tag == "space")
				{
					right = hit.collider.gameObject.GetComponent<SpaceScript>();
				}
			}

			adjacentsFound = true;
		}

		void FindFault(SpaceScript s, int p1Index, int p2Index)
		{
			if( s != null && s.used )
			{
				PiecesScript p1 = game.GrabPiece(s.pieceIDOnMe);
				PiecesScript p2 = game.GrabPiece(pieceIDOnMe);

				//make sure there where no nulls returned
				if(p1 == null || p2 == null)
				{
					print ("ERROR - SPACESPCRIPT - FINDFAULT - GRABPIECE() RETURNED NULL");
					return;
				}
				if( p1.connectionArray[p1Index] != p2.connectionArray[p2Index] || p1.connectionArray[p1Index] ==
ConnectionType.None || p2.connectionArray[p2Index] == ConnectionType.None) //right only using single and none
				{
					haveFault = true;
				}
			}
		}

		void FindExposed()
		{
			PiecesScript p = game.GrabPiece(pieceIDOnMe);
			if( p == null )
			{
				print ("ERROR - SPACESPCRIPT - FINDEXPOSED - GRABPIECE() RETURNED NULL");
				return;
			}
			if( top != null )
			{
				exposedCount = 0;
				if( p.connectionArray[0] != ConnectionType.Single )
				{
					exposedCount++;
				}
			}
		}
}
```

Code Sample

Jason Ege

Here are some code examples that, among others snippets, I am particularly proud of. There is one from each prototype including my portion of the sidescroller prototype (shooting).

Here are two snippets of code from the Ninja Spirit spiritual successor game:

```
//The following code demonstrates an ability to fire a partial spiral of bullets, or a "spread".
void EnemyFireSpreadOfBullets()
 {
 if (eBulletAmount > 0)
 {
 if (eBulletDelta > eBulletDeltaLimit)
 {
 Instantiate (eBullet, Player.transform.position, Quaternion.Euler (0,0,
eBulletRot));
 eBulletDelta = 0;
 eBulletRot += eBulletRotAdd;
 eBulletAmount--;
 }
 }
 if (eBulletAmount <= 0)
 {
 enemyfiringspread = false;
 eBulletAmount = eBulletAmountInit;
 eBulletRot = eBulletRotInit;
 eBulletDelta = 0;
 }
 eBulletDelta += Time.deltaTime;
 }
```

```
//The following code demonstrates firing a spiral of bullets that rotates around the player as viewed
from the side of the player. This is triggered elsewhere in the code by a keyboard button press.
void FireSpiralOfBullets()
 {
 int shotNum = 25;
 if (pBulletSpiralDelta > 0.025f)
 {
 //Duplicate the master bullet
 Instantiate(pBullet, Player.transform.position, Quaternion.Euler(0, 0,
pBulletSpiralRot));
 pBulletSpiralRot += 180.0f/shotNum; //Increase the rotation angle.
 pBulletSpiralDelta = 0; //Reset the delay between bullets timer.
 }
 pBulletSpiralDelta += Time.deltaTime; //Add to the delay timer.
 }
```

Here is a code snippet from the racing car game I made with Tyler

```
//This built-in Unity function catches when a trigger enter happens and acts based on it. This particular
section prevents the player from passing the finish line and then going back and forth across the finish
line. They must go through two other checkpoints that are approximately spaced at around 1/3 of the
way through the track and about 2/3 of the way through the track.
void OnTriggerEnter(Collider c)
 {
 if (c.name == "FinishLine") //If the player hits the finish line box collider
 {
 if (activecheckpoint == 0 || activecheckpoint == 3) //if the player has gone
through the last checkpoint before the finish line.
 {
 activecheckpoint = 1; //Reset checkpoint number.
 laps++; //Increase the lap counter.
 }
 }
 if (c.name == "Checkpoint1" && activecheckpoint == 1) //If the player hits the first
checkpoint collider and they have hit the finish line first.
 {
 activecheckpoint = 2; //Change the active checkpoint.
 }
 if (c.name == "Checkpoint2" && activecheckpoint == 2) //if the player hits the second
checkpoint collider and the player has hit the first checkpoint.
 {
 activecheckpoint = 3; //Set the active checkpoint to the last one.
 }
 }
```

Here is a code snippet from the Dragon Hunt game I made with Shaq.

```
//This is the bat's movement pattern. He cycles up and down slowly while flying in circles moving closer
and farther from the player.

float speed;
void Update() {
speed += initspeed;
 transform.rotation = new Quaternion (0.0f, transform.rotation.y, transform.rotation.z,
transform.rotation.w);
 transform.Translate (Mathf.Cos (speed/20) / 4.5f, Mathf.Sin (speed) / 4.5f, Mathf.Sin
(speed/20) / 4.5f);
 transform.rotation = new Quaternion (0.75f, transform.rotation.y, transform.rotation.z,
transform.rotation.w);
}
```

Here is a code snippet from the X-Ray game I made with Vinessa and Tyler.

```
//This code block handles switching between active layers and the color interpolation between a clear
texture and the crate texture gives the crates a see-through appearance without culling them out of
view entirely.
void MakeCratesSeeThrough(int seethroughlevel)
 {
 if (seethroughlevel == 2)
 {
 g_seethroughlevel = 2;
 for (int i = 0; i < Crates.Length; i++)
 {
 Crates[i].renderer.material.Lerp (ClearMaterial, CrateMaterial, 0.15f);
 }
 }
 else if (seethroughlevel == 1)
 {
 g_seethroughlevel = 1;
 for (int i = 0; i < Crates.Length; i++)
 {
 Crates[i].renderer.material.Lerp (ClearMaterial, CrateMaterial, 0.55f);
 }
 }
 else if (seethroughlevel == 0)
 {
 g_seethroughlevel = 0;
 for (int i = 0; i < Crates.Length; i++)
 {
 Crates[i].renderer.material = CrateMaterial;
 }
 }
 }
```