Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут»

Кафедра АСОІУ

ЗВІТ

про виконання лабораторної роботи № 3

з дисципліни

«Мобільно-орієнтована розробка програмного забезпечення»

|  |  |  |
| --- | --- | --- |
|  | Виконав: |  |
|  | Михайленко О.М.  Савченко В.В.  Бузинний А.М. |  |

Київ 2017

**Кэширование**

При разработке на Unity, особенно под мобильные устройства, не стоит злоупотреблять дорогими для производительности операциями, такими как, например, получение компонентов. Для этого стоит использовать [кэширование](http://www.jerrodputman.com/2014/03/27/quick-component-caching-in-unity/).

Если компонент требуется использовать не единожды, традиционный подход — объявить в скрипте, где он будет использоваться, переменную для него, взять нужный компонент один раз и в дальнейшем использовать полученное значение. Например:

Rigidbody rigidbody;

void Start ()

{

rigidbody = GetComponent<Rigidbody>();

}

rigidbody.velocity = transform.forward \* speed;

Кэширование при инициализации актуально также и для свойств, предоставляемых GameObject по умолчанию, таких как .transform, .render и других. Для доступа к ним явное кэширование все равно будет быстрее (да и большая часть из них в Unity 5 помечена как deprecated, так что хорошим тоном будет отказаться от их использования).

Transform thisTransform;

public override void Go (Transform purpose)

{

thisTransform.LookAt(purpose);

thisTransform.eulerAngles = new Vector3(0f, transform.eulerAngles.y, 0f);

gameObject.SetActive(true);

}

private void Start()

{

thisTransform = this.transform;

}

Также, по ходу нашей игры создается и уничтожается множество однотипных объектов. Гораздо рациональней скрывать их и просто использовать повторно. Для этого был использован пул объектов.

class ObjectPool : MonoBehaviour

{

public GameObject prefab { get; set; }

private List<GameObject> cache;

int maxInstances { get; set; }

public void Push(GameObject obj)

{

if (cache.Count < maxInstances)

{

obj.SetActive(false);

cache.Add(obj);

}

else

{

Destroy(obj);

}

}

public GameObject Pool(Vector3 position, Quaternion rotation)

{

foreach (GameObject i in cache)

{

if (i && i.activeInHierarchy == false)

{

i.SetActive(true);

i.transform.position = position;

i.transform.rotation = rotation;

return i;

}

}

GameObject obj = Instantiate(prefab, position, rotation) as GameObject;

cache.Add(obj);

return obj;

}

public ObjectPool(GameObject newPrefab, int newMaxInstances = 20)

{

maxInstances = newMaxInstances;

prefab = newPrefab;

cache = new List<GameObject>();

for (int i = 0; i < maxInstances; i++)

{

GameObject obj = Instantiate(prefab) as GameObject;

obj.SetActive(false);

cache.Add(obj);

}

}

public ObjectPool(){ }

}

Объекты создаются по указанному шаблону в указанном количестве и создаются еще только в том случае, если в пуле нет незадействованных объектов. После использования объекты не уничтожаются, а скрываются, после чего снова доступны для использования.

Пример использования в коде:

void Start ()

{

objectPool = new ObjectPool[planePrefabs.Length];

for (int i = 0;i < planePrefabs.Length; i++)

{

objectPool[i] = new ObjectPool(planePrefabs[i], 4);

}

}

void Spawn()

{

int rnd = Random.Range(0, planePrefabs.Length);

GameObject plane = objectPool[rnd].Pool(new Vector3(Random.Range(0, 20) - 10f,

7f, -15f), Quaternion.identity);

EvContr.OnSpawn(plane.GetComponent<Plane>());

}