сторонним, при уходе блокировать доступ к компьютеру.

Рис. 3. Административные способы защиты информационной системы аэропорта

- Необходимо регулярно обновлять антивирусное ПО (программное обеспечение).
- Для безопасного входа в сеть следует установить двухфакторную аутентификацию при ее наличии.
- Важно периодически менять пароль, используемый пользователем.
- Необходимо установить контроль доступа к информационной системе.

Заключение

Серьезными угрозами системы обработки информации аэропорта являются в равной степени и противоправные действия сотрудников, и взламывания правонарушителями корпоративной сети. Чтобы уменьшить риски несанкционированной аутентификации в информационную систему аэропорта, сотрудникам стоит применять программно-аппаратные и административные способы защиты информации. Существует множество рекомендации о том, как обеспечить безопасность корпоративных систем, однако злоумышленники придумывают новые способы взлома информационных систем, поэтому следует регулярно улучшать способы защиты информации в гражданской авиации.

Список литературы

1. **Мельников В.П.** Информационная безопасность и защита информации: учебное пособие / В.П. Мельников, С.А. Клейменов, А.М. Петраков — М.: Академия, 2008. — 336 с.
2. **Духан Е.И.** Программно-аппаратные средства защиты компьютерной информации. Практический курс: учеб. пособие / Е.И. Духан, Н.И. Синадский, Д.А. Хорьков — Екатеринбург: УрГУ, 2008. — 240 с.
3. **Внуков А.А.** Защита информации: учеб. пособие / А.А. Внуков — М.: Юрайт, 2018. — 261 с.

РАЗРАБОТКА АЛГОРИТМА ОПТИМИЗАЦИИ ИГРОВОГО ПРОЕКТА ПРИ ПОМОЩИ ШАБЛОНАПРОЕКТИРОВАНИЯ OBJECT POOL

Алтунин Андрей Владимирович
Студент 4-го курса Государственного технологического университета «СТАНКИН»

Бритвина Валентина Валентиновна
Кандидат педагогических наук, доцент кафедры «Управление и информатика в технических системах» Московского государственного технологического университета «СТАНКИН», доцент кафедры «Инфокогнитивные технологии» Московского политехнического университета

Аннотация. В статье рассмотрены различные способы оптимизации игровых проектов. В рамках рассмотренных способов, разработан и внедрен собственный алгоритм оптимизации, основанный на шаблоне проектирования object pool.

Ключевые слова: информационные технологии, оптимизация, игровые объекты, алгоритм.

Annotation. The article discusses various ways to optimize game projects. Within the considered methods, developed and implemented its own optimization algorithm based on the design pattern object pool.

Keywords: information technology, optimization, game objects, algorithm.

Введение

При разработке игр для мобильных устройств на базе Android и IOS, разработчики часто сталкиваются с проблемой оптимизации своих проектов. В первую очередь это связано с тем, что большинство современных смартфонов и планшетов недостаточно производительны, а значит забота о ресурсах, потребляемых игрой, должна быть на первом месте. Чем выше FPS в игровом проекте, тем выше его качество. Для решения проблем оптимизации существует масса различных способов.

Цель исследования

Разработать алгоритм оптимизации с применением шаблона проектирования object pool

Задачи исследования

- Анализ вариантов оптимизации мобильных игр, созданных при помощи игрового движка Unity.
- Разработать способы оптимизации проекта
- Экспериментально проверить эффективность способов оптимизации.

Анализ вариантов оптимизации мобильных игр

Вариантов оптимизации существует огромное количество, дело в том, что разные проекты требуют разных подходов. Однако, любая оптимизация начинается с графики.

Существуют следующие способы масштабирования потребления ресурсов которые представлены на рис. 1

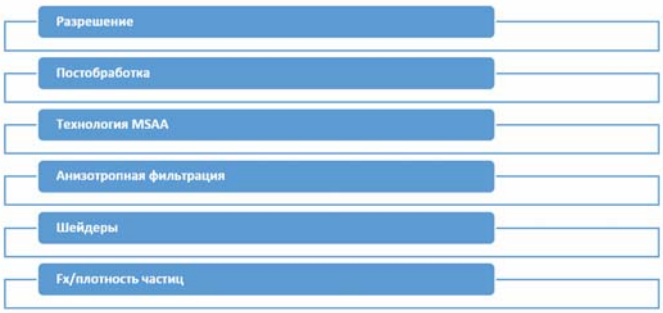


Рис. 1. Способы масштабирования потребления ресурсов

Производительность графики напрямую зависит от геометрической сложности объектов, а именно от количества вершин у моделей (полигонов). Для уменьшения нагрузки на GPU, используется подход Occlusion culling – отброс загороженной геометрии, когда движок не ренедрит объекты, загороженные другими объектами. В компьютерной 3D графике это не происходит автоматически. Чаще всего сначала отрисовываются объекты, расположенные дальше от камеры и уже поверх них отрисовываются ближние к камере объекты (это называется "overdraw"). Occlusion Culling отличается от Frustum Culling. Frustum Culling отключает только рендеринг объектов, не попадающих в область обзора камеры, не трогая при этом скрытые по overdraw объекты.

На мобильных устройствах отрисовка кадра напрямую привязана к скорости заполнения (скорость заполнения = пиксели экрана * сложность шейдера * перерисовка) и сложные шейдеры зачастую потребляют очень много ресурсов. В таком случае задействуются специальные мобильные шейдеры, они значительно экономят ресурсы за счёт своей простоты, хотя это и снижает красоту картинки.

Можно выделить общие правила для повышения эффективности использования вычислительных ресурсов:

- Используйте маленькое количество уникальных материалов, таким образом, движку будет легче производить отрисовку кадра.
- Используйте атласы текстур (большие изображения, содержащие коллекцию подизображений) вместо нескольких отдельных текстур. Это позволяет быстрее подгружать новые текстуры.
- Используйте средства визуализации shared.Material вместо Renderer.Material при использовании Атласов текстур и общих материалов.
- Старайтесь избегать ситуаций, когда несколько источников света освещают один объект.
- Уменьшайте общее количество проходов шейдера (тени, пиксельное освещение, отражения).
- Старайтесь как можно меньше использовать постобработку, это слишком затратно для мобильных устройств.
- Устанавливайте свойство Static для неподвижных объектов. Это называется – static batching, в таком случае движок отрисует объект только при загрузке сцены.
- FindObjectsOfType – довольно медленный метод, поэтому лучше избегать его частого использования
- Используйте сжатие текстур.

Зачастую, игра не использует в полной мере мощность процессора, в большей степени загружая GPU. Таким образом, часто разумно вытащить некоторую работу с GPU и поместить ее на CPU. Например отрисовка маленьких объектов, частичное обновление геометрии объектов, и mesh skinning (процесс связывания костей с вершинами меша).

Разберём ещё один подход, который поможет значительно сохранить ресурсы, если в игре часто создаются и удаляются одновременно небольшое количество объектов (например пули/ враги/ препятствия).Дело в том, что аллокация и деалокация памяти очень затратная функция, это связано с тем, что большинство скриптовых языков имеют автоматическое управление памятью в виде инструмента под названием – "сборщик мусора". Удаление ссылок на неиспользуемые ресурсы и последующие их освобождение происходит не сразу, а с определённым интервалом. В момент, когда сборщик мусора начинает свою работу, игрок может наблюдать падение FPS. Для решение такой проблемы был придуман порождающий шаблон проектирования – объектный пул (англ. object pool). Суть которого заключается в том, что новые объекты не создаются с нуля, а берутся уже инициализированные и готовые к использованию из некоторой базы. Когда

объект больше не используется, он не уничтожается, а помещается обратно в пул и ждёт очередного вызова. У такого подхода наблюдается серьёзный минус, заключающийся в необходимости постоянно расходовать ресурсы оперативной памяти.

Разработка и внедрение способов оптимизации

Попробуем написать собственный object pool и внедрить его в существующий проект. Коротко о проекте — игра в стиле tower defence. С определенным интервалом на сцене создаются следующие объекты: враги, снаряды. Задача — минимизировать вызов процедуры object.Instantiate.

Реализуем пул объектов.

Реализация

Пул будет сделан для следующих игровых объектов:

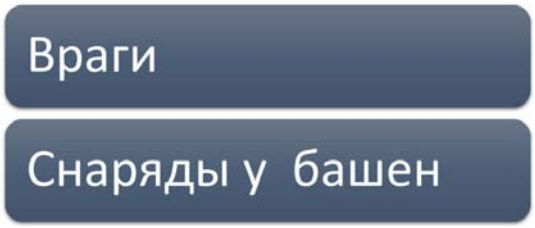


Рис. 2. Игровые объекты, для которых необходим объектный пул

Именно эти игровые объекты часто создаются и уничтожаются, а значит потребляют много ресурсов процессора.

Рис. 3. Компоненты объектного пула

Реализуем два компонента:

ElementPoolController.cs служит для добавления объекта в пул, то есть аналог метода Destroy. Все компоненты игрового объекта отключаются, а сам объект скрывается.

Листинг кода:

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

public class ElementPoolController : MonoBehaviour
{
    /// <summary>
    /// Analog of destroy. All components are disabled, object hiding.
    /// </summary>
    public void AddInPool()
    {
        gameObject.SetActive(false);
    }
}
```

Достаточно всего одного метода — AddInPool, при вызове которого происходит деактивация объекта, на котором висит данный скрипт.

Рассмотрим следующий скрипт PoolHierarchy.cs — этот скрипт управляет объектами пула.

Листинг кода:

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

public class PoolHierarchy : MonoBehaviour {

    private List<GameObject> elementsPool = new List<GameObject>();

    public GameObject InstantiateObj(GameObject prefab, Vector3 positionSpawn,
Quaternion rotation)
    {
        GameObject returnedObj=null;

        foreach(GameObject obj in elementsPool)
        {
            if (obj.activeInHierarchy == false)
            {
                returnedObj = obj.gameObject;
                returnedObj.transform.SetPositionAndRotation(positionSpawn,rotation);
                returnedObj.SetActive(true);
                break;
            }
        }

        if(returnedObj==null)
        {
            returnedObj = GameObject.Instantiate(prefab, positionSpawn, rotation);
            elementsPool.Add(returnedObj);

        }
        return returnedObj;
    }
```

```
    public void RemoveAllElements()
    {
        for(int i=0;i<elementsPool.Count;i++)
        {
            Destroy(elementsPool[i]);
        }
        elementsPool.Clear();
    }
}
```

Рассмотрим методы данного класса. Их всего два — InstantiateObj и RemoveAllElements. Первый метод является самым главным. Это тот же самый gameobject.instantiate (создание нового объекта на сцене), но с тем отличием, что объект создается не из некого префаба с нуля, а возвращается из пула элементов, которые находятся в List<GameObject> elementsPool. Принцип работы функции: для начала выполняется проход по всем элементам списка, и проверяется, что элемент в пуле не активен. В случае выполнения этого условия, важно установить координаты объекта в позицию, переданную как параметр функции. После чего, возвращаем нужный объект и выходим из цикла. Если же в списке нет элементов, удовлетворяющих условию, происходит обычное создание объекта на сцене из префаба.

Метод RemoveAllElements просто очищает пул, уничтожая его объекты.

Применение данных скриптов следующее. На объекты, работающие с пулом, необходимо прикрепить ElementPoolController.cs. Далее, создается объект, который будет выступать в качестве хаба, на него прикрепляется компонент PoolHierarchy.cs. Как пример: хабом для врагов будут спаунеры, а хабом для снарядов будут сами башни. Теперь удаление объектов будет происходить через контроллер, после чего объект будет записываться в пул. А компоненты генерации будут обращаться к пулу за новыми объектам.

Как видно, при данном подходе необходимо создавать свой пул для каждого типа объектов. При большом количестве типов, такой способ будет довольно громоздким и неудобным. Важное

замечание — большое количество таких пулов сильно загрузит оперативную память.

Заключение

После применения данного способа оптимизации удалось получить ~ + 6.6 fps. Главной характеристикой такого подхода оптимизации является повышение стабильности fps, что видно из рис. 4–5.

Рис. 4. До использования пула

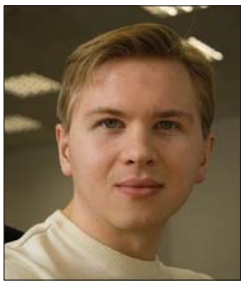
Рис. 5. После использования пула

Список литературы

1. Документация компании разработчика движка Unity3D [Электрон. ресурс]. — Режим доступа: https://docs.unity3d.com.
2. Alexander Shvets «Dive Into Design Patterns».
3. http://it.mmcs.sfedu.ru/wiki/Пул_объектов_(Object_Pool).

ПО ПРОЕКТИРОВАНИЯ КСЗИ ДЛЯ ОБЪЕКТА ЗАЩИТЫ

Тихомиров Александр Олегович
Студент 5-го курса направления «Информационная безопасность автоматизированных систем», Московского политехнического университета



Гневшев Александр Юрьевич
Старший преподаватель кафедры «Информационная безопасность» Московского политехнического университета

Аннотация. Данная статья посвящена проблеме разработки комплексной системы защиты информации для объекта защиты, а конкретнее оптимизации данного процесса. Проанализированы зарубежные и отечественные аналоги. Разработан подбор элементов защиты информации при проектировании комплексной системы защиты информации для объекта защиты. Сформирована база знаний уязвимостей и необходимых рекомендаций по защите информации на объекте.

Ключевые слова: КСЗИ, уязвимость, объект защиты, угроза, средство защиты, помещение, ПО, технические средства защиты, жизненный цикл.

Annotation. This article is devoted to the problem of developing an integrated information security system for a protected object, and more specifically to optimizing this process. Foreign and domestic analogues are analyzed. The selection of information security elements in the design of an integrated information security system for a protected object has been developed. A knowledge base of vulnerabilities and necessary recommendations for protecting information at the facility has been formed.

Keywords: CSIS, vulnerability, object of protection, threat, means of protection, premises, software, technical means of protection, life cycle.