СПОСОБЫ ОРГАНИЗАЦИИ УПРАВЛЕНИЯ ВНУТРЕННЕЙ ПАМЯТИ ПРИ РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ПЛАТФОРМЕ *UNITY*

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Турчин А. Ч., Трофимович А. В

Шаталова В. В. – канд. техн. наук, доцент
Хорошко В. В. – канд. техн. наук, доцент

В данной статье описаны способы организации управления внутренней памятью устройства, на котором выполняется программное обеспечение, созданное при помощи платформы *Unity.*

При разработке программного обеспечения всегда стоит помнить о проблеме утечки памяти. Именно такие проблемы в последствии могут стать очень трудно улавливаемыми, а на их устранение может уйтм не мало времени и ресурсов. Каждый язык программирования предоставляет собственные способы для управления внутренней памятью компьютера.

Разработка программного обеспечения на платформе *Unity* ведется при помощи языка программирования *C#,* код которого выполняется в среде исполнения *mono* [1]. Данная среда исполнения обладает механизмом автоматического управления памятью, который работает по схожему принципу, как и в среде *CLR*. При создании объекта, строки или массива, память для его хранения выделяется из центрального пула, который называется куча. Когда использование элемента прекращается, память, которую он занимал, можно будет освободить и использовать для чего-нибудь еще. В прошлом, выделение и освобождение этих блоков памяти с помощью вызовос соответствующих методов в основном лежало на плечах разработчиков.

Менеджер памяти отслеживает зоны в куче, которые определены как неиспользуемые. При запросе нового блока памяти, например, при создании нового экземпляра объекта, менеджер выбирает неиспользуемую зону, из которой следует выделить блок, и затем удаляет выделенную память из зоны известного неиспользуемого простарнства. Последующие запросы обрабатываются тем же способом, пока в неиспользуемой зоне будет достаточно места для выделения блока необходимого размера. Доступ к ссылочному элементу в куче может быть получен только пока есть ссылочные переменные, которые могут его найти. Если все ссылки к блоку памяти пропадут, то занимаемая им память может быть еще раз безопасно выделена. Чтобы определить, какие блоки кучи больше не используются, менеджер памяти просматривает все активные ссылочные переменные и отмечает блоки, к которым онисылаются как *live.* В конце поиска, любое пространство между используемыми блоками менеджером считаются пустым и в будущем может быть использовано для выделения. Такой процесс обнаружения и освобождения неиспользуемой памяти известен как сборка мусора [2].

При создании программного обеспечения на платформе *Unity* разработчик, как правило, имеет дело с игровыми объектами, которые отображаются на игровой сцене. Поэтому в данном случае есть делать, которую необходимо помнить. Для отображения объектов в *Unity* ссылки на все эти объекты должны присутствовать в графе сцены.

Граф сцены – структура данных, используемая в основном в векторных графических редакторах, а также игровых движках, и каждый узел данного графа является объектом некоего класса [3].

Соответственно, даже после удаления всех ссылок на конкретный игровой объект в коде на него все равно будет ссылаться граф сцены, что сделает автоматическое удаление невозможным. Поэтому в *Unity* существует метод *Destroy(),* который заставляет игровой движок удалять объект из графа сцены и при следующей сборке мусора, выделенная память будет освобождена.

При разработке игровых приложений на платформе *Unity* существуют несколько подходов для организации сборки мусора:

- маленькая куча с быстрой и частой сборкой мусора – эта стратегия лучше всего подходит игровым приложениям с долгими сессиями игрового процесса;
- большая куча с медленной, но не частой сборкой мусора – эта стратегия лучше всего подходит для игр, где выделения памяти относительно редки.

Таким образом, в данной статье были описаны основные стратегии организации управления внутренней памяти устройства, при разработке на платформе *Unity*. Были описаны нюансы данного игрового движка при управлении памятью.

Список использованных источников:

1. Среда исполнения *Mono* [Электронный ресурс]. – Режим доступа https://ru.wikipedia.org/wiki/Mono

2. Автоматическое управление памятью на платформе *Unity* [Электронный ресурс]. – Режим доступа : https://docs.unity3d.com/ru/current/Manual/UnderstandingAutomaticMemoryManagement.html