УДК 004.946

С.А. Зори, канд. техн. наук, докторант
А.П. Бровкина, магистрант
Т.А. Кандиболоцкая, магистрант
Донецкий национальный технический университет, г. Красноармейск, Украина
sa.zori1968@gmail.com
anegka@hotmail.com
tanya.kandibolotskaya@mail.ru

Повышение реалистичности и оптимизация моделируемых процессов симулятора «Виртуальная шахта»

В работе рассматривается реализация динамики взаимодействия оператора и объектов окружения для повышения реалистичности моделирования в интерактивном симуляторе «Виртуальная шахта» для предприятия горнодобывающей промышленности. Рассмотрено создание анимации 3D-объектов моделируемой среды. Применены способы оптимизации системы и исследовано их влияние на эффективность работы симулятора. Данная система позволяет моделировать опасные для жизни работников шахты ситуации и осуществлять тренинг в режиме реального времени.

Ключевые слова: симулятор, шахта, динамика взаимодействия, оператор, визуализация, моделируемая среда

Введение

Интенсивное развитие аппаратных и программных средств компьютеризации позволяет все шире использовать компьютерные системы в различных областях, которые включают в себя тренинг персонала. Ради подготовки и повышения квалификационного уровня специалистов создается программное обеспечение, которое облегчает и совершенствует данный процесс [1].

Проект «Виртуальная шахта» является интерактивным симулятором технических объектов и технологических процессов на предприятии горнодобывающей промышленности. Данный симулятор может использоваться для ознакомления с шахтной средой и оборудованием, а также позволяет получить сотрудников с более высоким уровнем подготовки и снизить риск травм и убытков в случае аварии. Подробное описание проекта приведено в [2]

Для повышения степени реалистичности созданы анимации объектов, а также созданы скрипты, которые позволяют управлять состоянием анимации и прототипа оператора, что позволит воссоздать поведение объектов в условиях, приближенных к реальным. Также были исследованы способы оптимизации проекта для увеличения качества отображения виртуальной среды. Все операции производились с помощью пакета для создания компьютерной графики Blender и инструмента разработки двух- и трехмерных приложений и игр – игрового движка Unity [3-5].

Описание разработки

После разработки прототипа шахты и 3D-объектов шахтного оборудования важным этапом является создание анимации созданных объектов, которая позволяет наглядно продемонстрировать работу оборудования. Unity предоставляет собственную систему анимирования объектов и персонажей. Для анимации объектов, созданных в Blender, объекты и анимация должны быть перенесены в Unity при помощи конвертирования в формат .fbx.

Система анимации Unity основана на концепции анимационных клипов, которые содержат информацию о том, как некоторые объекты или их составляющие должны изменить позицию, вращение, или другие свойства, в том числе свойства присоединенных скриптов, в течение времени. Каждый ролик можно рассматривать как одну линейную запись с набором ключей анимации.

Анимационные клипы являются ключевыми объектами при создании полноценной анимации объекта в Unity и составляют в совокупности всю полосу анимации, разбитую на отдельные участки. Они представляют собой отдельные элементы движения, такие как, например, поднятие ноги при перемещении, поднятие руки, изменение свойств или перемещение в пространстве до определенного момента, и могут быть объединены разными способами, образуя живые и реалистичные анимации.

Создание анимации проходит в несколько этапов:

1) Создается сам файл с анимацией;
2) Выбирается объект (часть объекта) который необходимо проанимировать;
3) Создаются кадры с точками остановки, и изменяется положение объекта (или его части).

Анимационные клипы организовываться в структурированную систему из блок-схем под названием Animator Controller. Animator Controller выступает в роли «Машины состояния», которая отслеживает, какие клипы должны играть в настоящее время, а когда анимация должна измениться или смешаться с другой [3].

Для увеличения реалистичности, для некоторых объектов создана анимация со звуком. Для создания динамики в симуляторе персонаж должен взаимодействовать с созданными объектами. Для этого объекты были проанимированы, но чтобы вызвать анимацию не сразу при запуске симулятора, а при определенных условиях, нужны триггер(событие) и скрипт, который производит взаимодействие его с персонажем.

Общая схема параметрической модели визуализации объекта, которая содержит все необходимые для взаимодействия с объектом компоненты, приведена на рис. 1.

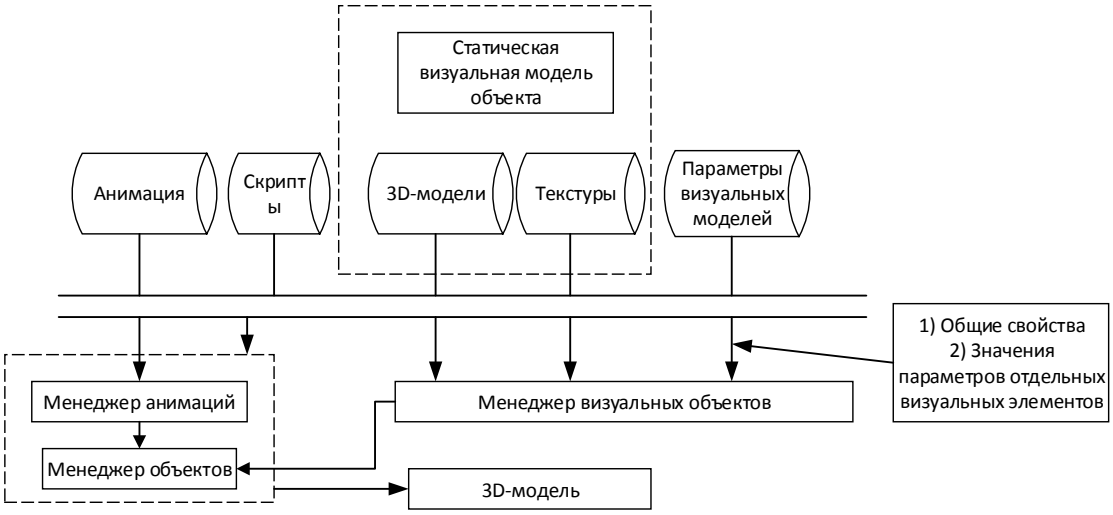

Рисунок 1 - Общая схема параметрической модели визуализации объекта

Триггер - механизм, проверяющий присутствие каких-либо объектов игрового мира в заданном пространстве и выполняющий некоторые запрограммированные действия при вызове связанных с триггером событий при помощи связанного с ним скрипта.

В ходе разработки проекта была смоделирована ситуация, которая демонстрирует последствия пренебрежения правилами безопасности, - слишком близкое нахождение к движущемуся механизму. Такая ситуация может привести к тяжелым увечьям, ампутации попавшей в механизм конечности и смерти.

Данный скрипт окрашивает экран в красный цвет путем обращения к скрипту камеры оператора, снижает визуально отображаемый уровень здоровья, а также воспроизводит человеческий крик, не только графически, но и визуально привлекая внимание оператора к внештатной ситуации (рис. 2). Работает данный скрипт только тогда, когда проверяемая им анимация воспроизводится, а оператор находится в зоне триггера, связанного с данным скриптом. Когда уровень здоровья равен нулю или меньше, симулятор перезапускает сцену. Данный скрипт использует универсальный подход, связанный с использованием триггера для регистрации столкновения оператора и объекта.

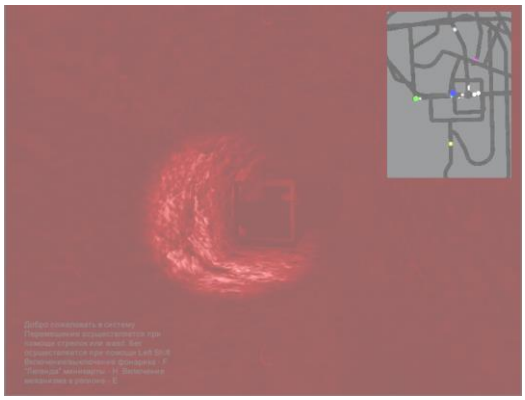

Рисунок 2 – Окрашивание экрана при возникновении внештатной ситуации

Так как объект состоит из множества простых Mesh, стандартные методы OnCollisionEnter() и OnControllerColliderHit [3] будут неэффективны, так как используются для объектов, состоящих из только одного Mesh. При использовании данных функций возникает проблема увеличения количества расчетов и обработки множества объектов, что приведет к сни-

жению производительности в несколько раз при росте системы и увеличении количества сложных объектов, с которыми можно взаимодействовать [4]. В этом случае имеет смысл поступиться оптимизацией в пользу простоты и универсальности: вместо просчета непосредственных столкновений с объектом использован триггер, с помощью которого проверяется один простой объект, а не множество сложных.

В качестве параметров в скрипт передается объект с анимацией, чье состояние проверяется, источник звука и объект-камера, которая окрашена красным.

```
public class ColliderHit : MonoBehaviour {
    //public GlobalFog script2Reference;
    public GameObject Object; //музыка
    public GameObject animObj; //анимация
    public int hitman=50;
    //public GameObject camObj; //камера
    // Use this for initialization
    void Start () {

    }

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

    }


    void OnTriggerEnter(Collider all)
    {
        if (all.gameObject.CompareTag ("MainCamera")) {
            if (animObj.animation.isPlaying) {
                // проиграть звук вскрика
                Object.audio.Play ();
                //окрасить камеру красным цветом
                NoiseEffect target = (NoiseEffect)FindObjectOfType(typeof(NoiseEffect));
                target.enabled=true;
                //уменьшить здоровье и постепенно уменьшать до нуля. После чего - ресет игры
                damage Dam = (damage)FindObjectOfType(typeof(damage));
                Dam.DamageHealth(hitman);
            }
        }

    }
    void OnTriggerExit(Collider all)
    {
        NoiseEffect target = (NoiseEffect)FindObjectOfType(typeof(NoiseEffect));
        target.enabled=false;
    }
```

Рисунок 3 - Код триггера, отвечающий за реакцию на столкновение

Выполнены анимации проходческого комбайна, лифта, угольного комбайна, спирального классификатора для воссоздания их работы в шахте в реальных условиях. Это дает возможность смоделировать ситуации, при которых оператор может быть травмирован (попасть под лифт, травмироваться при попадании под движущиеся части механизма).

Оптимизация проекта и исследование эффективности

В процессе разработки игр очень часто возникают проблемы со сложными детализированными 3D-моделями. Основным недостатком таких моделей являться необходимость выделения для них большого объема вычислительных ресурсов, а также сложность их обработки. Также наличие в сцене или модели большого числа полигонов ограничивает возможность запуска созданного приложения на устройствах с низкой вычислительной мощностью. В итоге одним из этапов моделирования объектов является их оптимизация.

Существуют несколько способов создания визуального представления трехмерного объекта с минимальными потерями качества. Все функции предоставляет графический редактор Blender.

1) Слияние соседних близких вершин. Очень часто модель может иметь несколько вершин находящихся на одной прямой или на малом расстоянии друг от друга.

2) Извлечение дубликатов. Функция позволяет упростить сеть посредством соединения вершин, которые находятся на расстоянии друг от друга, более близком, чем определено параметром. Пример использования функции извлечения дубликатов показан в таблице 1.

3) Создание новой меш-топологии. Существуют модификаторы, позволяющие «сгладить» объект тем самым убирая некоторые полигоны.

Таблица 1 – Использование опции Извлечение дубликатов.

Объект	Начальное количество вершин	Количество вершин после использования Remove Double
Скребковый конвейер	68427	66603
Проходческий комбайн	83923	82953
Автономная газовая защита	11353	11037

Был опробован вариант управления в виде от третьего лица. Такой вид называется Third Person Player (TPS). Однако такой подход, после перенесения всех скриптов для камеры и взаимодействия с персонажем, оказался неэффективен: понижение FPS составило в состоянии покоя (симулятор запущен, анимации воспроизводятся, однако ни одна клавиша не нажата) с 22 до 10, а в состоянии передвижения в любую из сторон, включая поворот мышью, - с 17 до 4. Поэтому в ходе разработки принято решения отказаться от реализации вида от третьего лица и остановиться на виде от первого лица с использованием модели рук оператора, которые будут показаны и анимированы в процессе для повышения реалистичности и создания эффекта присутствия в среде.

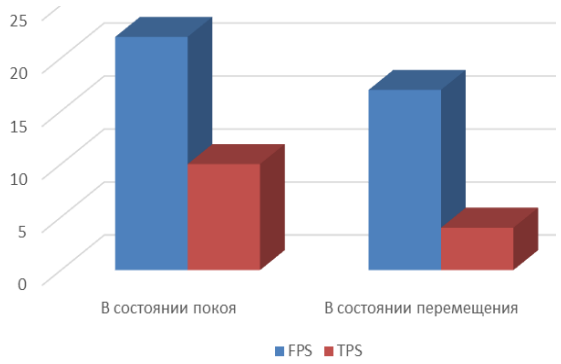

Рисунок 4 – Диаграмма количества кадров в секунду для FPS и TPS

В связи с тем, что при увеличении сложности проекта (добавление высокополигональных моделей, анимирование, добавление скриптов, триггеров и объектов взаимодействия) количество кадров в секунду падает, необходимо определить, какие события и ситуации могут привести к уменьшению FPS и каким образом можно увеличить быстродействие симулятора без потери реалистичности. Проанализированы настройки графики и их влияние на производительность на графическом процессоре Intel®. Тестирование проводилось на компьютере с процессором Intel® Core™ 3-го поколения с ГП Intel® HD Graphics 4000.

В Unity доступны дополнительные параметры рендеринга, влияющие на производительность. Для анализа возможностей оптимизации в Unity опробованы изменения настроек качества и проанализировано изменение количества FPS.

1) Было проанализировано влияние качества рендеринга текстур на производительность Доступны разрешения 1/8, ¼, ½ или полное разрешение. Для оценки производительности при разном разрешении текстур измерена кадровая скорость при всех доступных в Unity настройках качества по умолчанию, изменяя только качество текстур перед каждым измерением. При уровне качества Fantastic (таблица 2) видно, что производительность незначительно изменилась при изменении размера текстур. Диаграмма 5 наглядно демонстрирует данную зависимость.

Таблица 2 - Изменение FPS при переключении между различным качеством текстур

Качество текстур	1/8	1/4	1/2	Full
Самое быстрое	151	151	150	150
Быстрое	165	164	163	161
Простое	155	153	151	151
Хорошее	130	130	130	128
Красивое	113	114	114	113
Фантастическое	72	73	72	69

Производительность существенно не изменяется в зависимости от размера текстур.

2) Shadow Distance — параметр, определяющий глубину отбраковки, используемую для теней объектов. Если объект находится в пределах заданного расстояния от камеры, то тени объекта отрисовываются, иначе тени такого исключаются из отрисовки. Тени могут отрицательно повлиять на производительность, поскольку их расчет и отрисовка являются ресурсоемкими операциями. Уровни качества Fastest и Fast не используются в тестировании, поскольку в этих режимах тени отключены. Результаты показаны в таблице 3. Наглядная демонстрация - на диаграмме 6.

Таблица 3- Изменение кадровой скорости при изменении Shadow Distance

Shadow distance	0	1	5	10	15	25	50
Простое	124	114	96	92	82	77	73
Хорошее	79	63	56	55	52	50	46
Красивое	39	35	34	33	32	30	28
Фантастическое	35	32	31	30	29	28	26

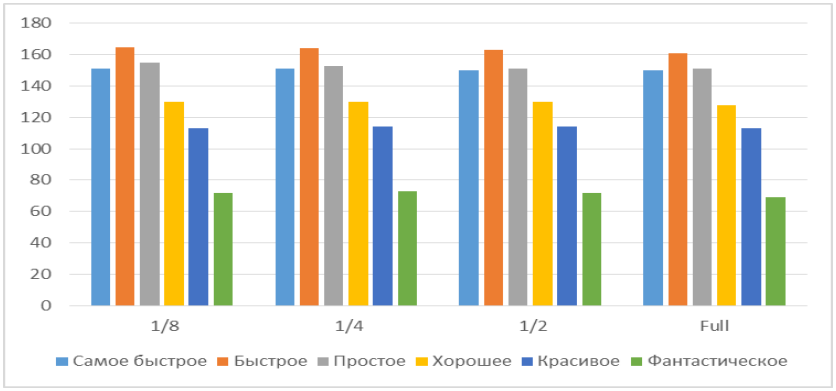

Рисунок 5 - Диаграмма зависимости качества текстур и ускорения

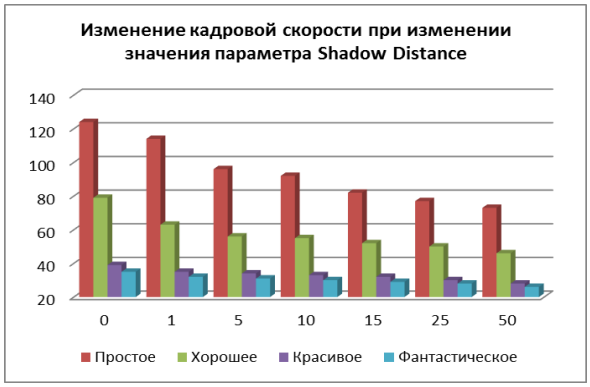

Рисунок 6 – Изменение FPS при изменении значения параметра Shadow Distance

3) Тени значительно влияют на производительность. Тест показал, что кадровая скорость упала почти вдвое при переключении расстояния с 0 до 50 в режиме Simple.

4) Расстояние отбраковки слоев. Всем объектам в Unity при создании назначается слой. Изначально всем объектам назначается слой по умолчанию, но можно создать собственные уникальные слои, что позволяет группировать объекты для дальнейшей обработки.

Камера не будет отрисовывать объекты, находящиеся за пределами плоскости отсечения в Unity. С помощью сценариев Unity можно задать для определенных слоев более короткое расстояние до плоскости отсечения.

Для тестирования производительности создано три сцены, заполненные объектами низкой, средней и высокой сложности. Объекты собраны вместе и расположены с постепенным удалением от камеры.

Измерение первое: полигонов — 278, вершин — 218. Измерение второе: полигонов — 4398, вершин — 4400. Измерение третье: полигонов — 112 074, вершин — 65 946.

В таблицах 4-7 показано изменение кадровой скорости в сценах.

Таблица 4 – Данные при отрисовке сцены с простыми моделями

Количество простых моделей	50	100	150	200	250	300
Самое быстрое	355	308	302	274	225	209
Быстрое	310	288	279	279	283	276
Простое	295	273	262	266	265	261
Хорошее	143	140	138	135	133	131
Красивое	79	77	76	75	74	73
фантастическое	71	69	68	68	67	66

Таблица 5- Данные при отрисовке сцены с моделями средней сложности

Количество средних моделей	5	10	15	20	25	30
Самое быстрое	329	285	277	263	243	212
Быстрое	295	256	246	236	210	202
Простое	288	250	240	230	214	191
Хорошее	142	134	130	121	119	111
Красивое	82	77	73	68	66	60
фантастическое	77	72	68	66	63	57

Таблица 6 - Данные при отрисовке сцены со сложными моделями

Количество сложных моделей	1	2	3	4	5	6
Самое быстрое	213	145	116	93	76	67
Быстрое	213	157	124	100	83	73
Простое	207	154	121	98	81	73
Хорошее	133	120	102	84	73	64
Красивое	69	49	39	32	26	23
фантастическое	66	47	37	31	25	22

В таблице 7 показано, как производительность изменяется при увеличении количества объектов на экране, особенно если это сложные объекты. Качество моделирования можно оценить на рисунке 7.

Таблица 7 - Данные всех измерений в режиме Fantastic

Количество изображенных объектов	50/5/1	100/10/1	150/15/3	200/20/4	250/25/5	300/30/6
Режим Fantastic						
Простые	71	69	68	68	67	66
Средние	77	72	68	66	63	57
сложные	66	47	37	31	25	22

Рисунок 7 - Данные измерений в режиме Fantastic

Использование layerCullDistances позволит значительно повысить производительность.

4) Графические эффекты. В Unity содержится ряд графических эффектов, которые улучшают вид сцены. Рассеянное затенение в экранном пространстве (SSAO) — это графический эффект в составе пакета Image. Итоговые изображения выглядят схоже, но производительность существенно различается. У сцены без SSAO кадровая скорость составила 32 кадра в секунду, а с SSAO — 24 кадра в секунду, то есть на 25 % ниже.

Следует соблюдать осторожность при добавлении графических эффектов, поскольку они могут отрицательно повлиять на производительность.

5) Исключение заслоненных объектов — это отключение рендеринга не только тех объектов, которые находятся за плоскостью отсечения камеры, но также и объектов, скрытых за другими объектами. Это очень выгодно с точки зрения производительности, поскольку значительно сокращается объем обрабатываемой информации. Можно добиться увеличения производительности более чем в 4 раза.

6) Уровень детализации (LOD). С помощью уровня детализации (LOD) можно присвоить одному объекту несколько моделей разной сложности и переключаться между ними в зависимости от расстояния между объектом и камерой. Это выгодно с точки зрения производительности для сложных объектов, находящихся далеко от камеры.

С помощью уровня детализации можно автоматически упрощать модели.

Для тестирования прироста производительности при изменении уровня детализации создана сцена с объектами, 3-х разных моделей. Измерена кадровая скорость при использовании моделей различной сложности. Результаты продемонстрированы в таблице 8.

Таблица 8 – Данные для измерений с объектами

Качество	Объект 1	Объект 2
Наилучшее – уровень 0	Вершин — 7065 Полигонов – 4999	Вершин — 5530 Полигонов – 3694
Среднее качество — уровень детализации 1	Вершин — 6797 Полигонов — 4503	Полигонов — 5476 Вершин — 3690
Низкое качество — уровень детализации 2	Вершин — 474 Полигонов – 308	Полигонов – 450 Вершин — 320

В таблице 9 показано повышение производительности при настройке и использовании разных уровней детализации. Кадровая скорость значительно возрастает при переключении на менее сложные модели.

Таблица 9 - Сравнение кадровой скорости при разных уровнях детализации

Уровень детализации	Уровень 0	Уровень 1	Уровень 2
Кадров в секунду	160	186	240

7) Пакетная обработка. Избыточное количество вызовов рендеринга может привести к чрезмерной нагрузке на ЦП и снижению производительности. В Unity поддерживается пакетная обработка, позволяющая поместить несколько игровых объектов в один вызов рендеринга. Статическая пакетная обработка предназначена для статических объектов, а динамическая — для движущихся объектов. Динамическая пакетная обработка выполняется автоматически, статическую пакетную обработку требуется задавать вручную. Для тестирования прироста производи-

тельности при статической пакетной обработке создана сцена со сложными объектами и измерена кадровая скорость с пакетной обработкой и без нее (таблица 10).

Таблица 10 - Кадровая скорость и количество вызовов рендеринга при пакетной обработке

Пакетная обработка статических объектов	Выкл	Вкл
Кадров в секунду	24	58
Вызовов отрисовки	5144	390

Для получения наибольшего преимущества от пакетной обработки следует объединить как можно больше объектов в пакет для одного вызова рендеринга.

Заключение

В работе для решения задачи по созданию анимаций объектов системы виртуальной реальности для интерактивных симуляторов и тренажеров на предприятиях горнодобывающей промышленности, проведен анализ принципов построения динамического взаимодействия между различными объектами, персонажем, а также с самой системой. Разработаны анимации 3D-моделей шахтного оборудования и создана основа для динамического взаимодействия между ними. Произведен анализ оптимизации моделируемых процессов для повышения производительности и качества визуализации и моделирования виртуальной сцены. Анализ показал, что рекомендации по оптимизации являются существенными для применения их к проекту и позволят в комплексе повысить скорость быстродействия симулятора за счет незначительного снижения качества текстур, меньшей детализации теней, оптимизации количества полигонов для сложных объектов, а также статической пакетной обработки объектов. Выработаны рекомендации, позволяющие оптимизировать виртуальный симулятор в процессе создания, что значительно уменьшает время на разработку.

Список использованной литературы

1. Мочалов П.С., Титов І.В. - Технології та результати створення інтерактивних тренажерів в 3D віртуальних середовищах, - УФУ .: м.Новокузнецьк, 2014. - 10 с.].
2. Зори С.А., Бровкина А.П., Кандиболоцкая Т.А. Интерактивный симулятор технических объектов и технологических процессов для предприятия горнодобывающей промышленности. Наукові праці Донецького національного технічного університету. Серія: "Інформатика, кібернетика та обчислювальна техніка". №1 (20)' 2015. С. 78-83
3. Мануал Unity[Интернет -ресурс]. – электронные текстовые данных. – Режим доступа: http://ocs.unity3d.com/ru/current/Manual/AnimationOverview.html
4. MonoBehaviour.OnControllerColliderHit (ControllerColliderHit). Unity Documentation. [В Интернете] [Цитировано: 17 10 2015 г.]http://docs.unity3d.com/ScriptReference/MonoBehaviour...
5. Unity3D First Person Controlller Collision detection. StackOverflow. [В Интернете] [Цитировано: 17 10 2015 г.]http://stackoverflow.com/questions/9480167/unity3d-fi..

Надійшла до редколегії 26.10.2015

С.А.ЗОРІ, А.П. БРОВКІНА, Т.О. КАНДИБОЛОЦЬКА

ПІДВИЩЕННЯ РЕАЛІСТИЧНОСТІ ТА ОПТИМІЗАЦІЯ ПРОЦЕСІВ МОДЕЛЮВАННЯ СИМУЛЯТОРА «ВІРТУАЛЬНА ШАХТА»

У роботі розглядається реалізація динаміки взаємодії оператора і об'єктів оточення для підвищення реалістичності моделювання в інтерактивному симуляторі «Віртуальна шахта» для підприємства гірничодобувної промисловості. Розглянуто створення анімації 3D-об'єктів змодельованого середовища. Застосовані способи оптимізації системи і досліджено їх вплив на ефективність роботи симулятора. Дана система дозволяє моделювати небезпечні для життя працівників шахти ситуації і здійснювати тренінг в режимі реального часу.

Ключові слова: симулятор, шахта, динаміка взаємодії, оператор, візуалізація, віртуальне середовище

S.A.ZORI, A.P. BROVKINA, T.A.KANDIBOLOTSKAYA

INCREASING REALISTIC AND OPTIMIZATION OF SIMULATED PROCESSES OF SIMULATOR "VIRTUAL MINE"

In this paper considers implementation of dynamics of operator and objects interaction to improve the modeling realistic in interactive simulator "Virtual mine" for mining companies.
Consider creating animation of 3D-objects and realization of interaction between the operator and the simulated environment. This system allows to get acquainted with the prototype of the mine, as well as to simulate the life-threatening situation of mine workers for training in real time.

Keywords: simulator, mine, dynamics of interaction, operator, visualization, virtual environment