СИСТЕМНЫЙ АНАЛИЗ И ОБУЧАЮЩИЕ СИСТЕМЫ
SYSTEMS ANALYSIS AND TRAINING SYSTEMS

УДК 004.853

Н. И. Гурин, Е. С. Сахонь
Белорусский государственный технологический университет

ТЕХНОЛОГИЯ РАЗРАБОТКИ СИМУЛЯТОРОВ ЛАБОРАТОРНЫХ УСТАНОВОК ДЛЯ ДИСТАНЦИОННОГО ОБУЧЕНИЯ

Статья посвящена описанию разработанного технологического стандарта по созданию симуляторов лабораторных установок для выполнения лабораторных работ студентами технических вузов в дистанционном режиме. Предложена технология разработки симуляторов на основе 3D-моделирования объектов в среде 3ds MAX с обеспечением их интерактивности в среде Unity на основе языка C#. Разработаны методы и приемы реалистичного представления и использования рабочих элементов симулятора, меню элементов управления на основе Canvas, общая схема управления скриптами и последовательностью выполнения эксперимента на установке с заполнением таблицы результатов и их анализом. Предложен технологический шаблон симулятора, который может быть использован для разработки симулятора любой лабораторной установки при условии внесения в предлагаемый шаблон соответствующих визуальных и функциональных особенностей установки, но без изменения ее базового программного кода и пользовательского интерфейса, что позволяет существенно сократить сроки разработки симуляторов для проведения лабораторных работ в дистанционном режиме.

Ключевые слова: дистанционное обучение, компьютерные обучающие системы, интерактивная графика, симуляторы лабораторных установок.

Для цитирования: Гурин Н. И., Сахонь Е. С. Технология разработки симуляторов лабораторных установок для дистанционного обучения // Труды БГТУ. Сер. 3, Физико-математические науки и информатика. 2021. № 1 (242). С. 48–53.

N. I. Gurin, E. S. Sahon
Belarusian State Technological University

TECHNOLOGY FOR THE DEVELOPMENT OF LABORATORY SIMULATORS FOR DISTANCE LEARNING

The article is devoted to the description of the developed technological standard for the creation of simulators of laboratory installations for the performance of laboratory work by students of technical universities in remote mode. The technology for developing simulators based on 3D modeling of objects in the 3ds MAX environment is proposed, ensuring their interactivity in the Unity environment based on the language of C#. Methods and techniques have been developed to realistically represent and use the simulator's work elements, canvas-based control menus, a general script management scheme, and a sequence of experiment execution on the installation with the filling of the results table and their analysis. Offered a technological simulator template, which can be used to develop a simulator of any laboratory installation, provided that the proposed template includes the appropriate visual and functional features of the installation, but without changing its basic code and user interface, which allows to significantly reduce the time of development of simulators for laboratory work in remote mode.

Key words: distance learning, computer training systems, interactive graphics, laboratory simulators.

For citation: Gurin N. I., Sahon E. S. Technology for the development of laboratory simulators for distance learning. *Proceedings of BSTU, issue 3, Physics and Mathematics. Informatics, 2021*, no. 1 (242), pp. 48–53 (In Russian).

Введение. Современные технологии дистанционного обучения широко применяются в различных учреждениях образования и обучающих центрах. Вместе с этим недостаточное развитие получила технология разработки симуляторов установок для проведения лабораторных работ в виртуальном режиме. Обучающая среда для дистанционного обучения в основном ограничивается текстовыми, графическими, анимационными и мультимедийными учебными материалами, представляемыми в электронном виде. Для полного охвата учебного процесса, проводимого в дистанционном режиме [1], такой обучающей среде обычно недостает симуляторов для выполнения работ на лабораторных установках, а также функции диалога обучаемого с виртуальным преподавателем, роль которого выполняет сама среда в онлайн режиме [2]. В работе представлена технология создания симуляторов установок для выполнения лабораторных работ студентами технических вузов в дистанционном режиме.

Имитационные модели динамических процессов, которые используются при создании таких симуляторов, позволяют имитировать реальные установки и объекты исследования, виртуально обеспечивают условия для проведения реального эксперимента, позволяют получить адекватные и реалистичные результаты. В принципе для любого реального динамического процесса методами имитационного моделирования можно разработать его имитацию на экране компьютера и полностью управлять его проведением. Другими словами, любая задача, в которой нужно учесть определенные закономерности, может быть представлена визуальной имитационной моделью поведения взаимосвязанных объектов, в которой исходные данные являются начальными значениями параметров взаимодействующих объектов, а их изменение будет немедленно отражаться в визуализации динамического процесса, отображаемого на экране компьютера в ходе его выполнения.

Основная часть. Подготовленный технологический стандарт предлагает готовые решения для следующих задач при разработке компьютерных симуляторов:

- разработки интуитивно понятного интерфейса для симуляторов лабораторного практикума;
- функционала динамических процессов при выполнении виртуальных лабораторных работ на симуляторе;
- сборки разработанного приложения для симулятора под разные платформы, используемые в дистанционном обучении.

Технологический стандарт разработан на основе межплатформенной среды Unity и языка программирования C# для управления интерактивными элементами симулятора, а также среды 3ds MAX для разработки и экспорта в Unity 3D-моделей элементов установки. Экспорт разработанных приложений обеспечен под веб-ресурсы и мобильные кросс-платформенные приложения для использования в дистанционном обучении и самостоятельной работе студентов.

Обучение с помощью данного ресурса может быть осуществлено:

- на сайте, на котором располагается полная версия КОС;
- персональном компьютере посредством скачивания (Desktop);
- с помощью мобильного приложения в виртуальной реальности.

Разработанный на основе стандарта веб-ресурс симулятора располагает полным функционалом для осуществления лабораторных работ в виртуальном режиме.

Выполнение лабораторной работы на симуляторе происходит аналогично действиям, осуществляемым при выполнении лабораторной работы на занятии в учебной аудитории с преподавателем. В первую очередь пользователю предлагается ознакомиться с теорией к данной лабораторной работе, при этом полная версия теоретического материала по данной тематике всегда доступна на веб-ресурсе симулятора установки. При наведении курсора мыши на отдельные элементы установки выводится информация о их назначении, а при щелчке мышью камера приближается к элементу для его рассмотрения в удобном ракурсе.

Ход выполнения лабораторной работы задается программно по этапам, последовательность которых указывается соответствующими подсказками в информационном окне. Причем лишь после выполнения задачи на определенном этапе эксперимента пользователь может приступить к следующей. После выполнения лабораторной работы на симуляторе все сделанные пользователем измерения результатов работы сохраняются в таблицу, которую можно экспортировать в отдельный файл.

Для использования шаблона симулятора под реализацию проекта симулятора конкретной лабораторной установки необходимо в проекте Unity шаблона выполнить следующие действия:

1) создать 3D-модели всех элементов данной установки и ее окружения в 3ds MAX и импортировать их на сцену в проект Unity вместо элементов шаблонной установки;

2) переименовать UI-элементы в проекте шаблона в соответствии с элементами данной лабораторной установки и добавить по аналогии недостающие элементы;

3) внести свое описание для каждого элемента в код класса ScriptableObject (лист. 1), хранящийся в ресурсах используемого шаблонного проекта Unity, который позволяет создавать в приложении объекты для хранения больших объемов общих данных, не зависящих от экземпляров скриптов;

4) внести соответствующие корректировки в используемые шаблонные скрипты для данной лабораторной установки, которые определяются ее функционалом.

Лист. 1. Код класса ScriptableObject

```
public class Info : ScriptableObject {
 [SerializeField]
 private string information;
 [SerializeField]
 private Sprite image;
 public string GetInfo()
 { return information; }
 public Sprite GetImage()
 { return image; }
}
```

Основные модули разработанного шаблона для симулятора лабораторной установки продемонстрированы на примере работы с симулятором лабораторной установки «Наблюдения колец Ньютона» (рис. 1). Цель этой лабораторной работы на данной установке заключается в том, чтобы, используя интерференцию света, определить в результате проведенных измерений радиусов колец Ньютона красного, желтого, зеленого и синего цветов радиус кривизны исследуемой линзы.

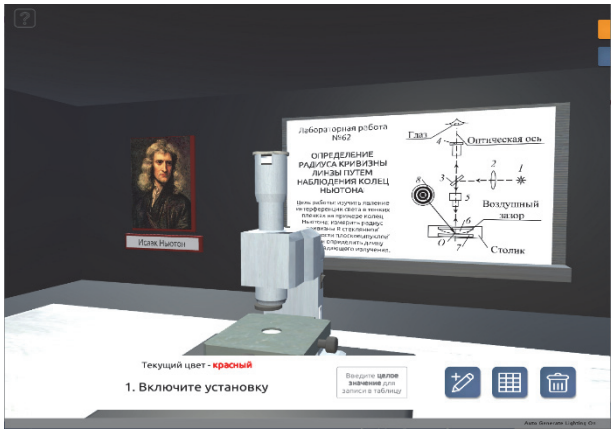

Рис. 1. Общий вид симулятора установки с его окружением на сцене

В первую очередь необходимо ознакомиться с установкой в разделе меню «УСТАНОВКА». Для этого следует просто нажимать на выпадающие пункты меню – будут подсвечиваться соответствующие части лабораторной установки (рис. 2).

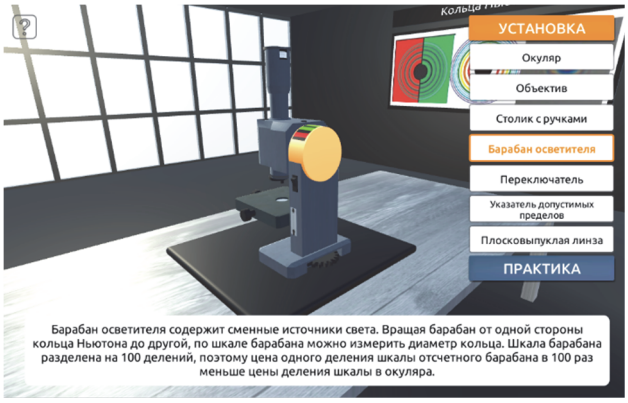

Рис. 2. Симулятор в режиме ознакомления с элементами установки

Для выполнения практической части лабораторной работы необходимо нажать мышью кнопку «ПРАКТИКА» и последовательно выполнять те действия, которые выводятся на информационную панель в нижней части экрана (рис. 1). При этом на данной панели присутствует окно для ввода измерений и кнопки занесения значений в таблицу, вывод таблицы на экран и очистка таблицы для проведения нового эксперимента.

Управление камерой реализовано стандартными средствами языка C# для среды Unity (лист. 2). Для перемещения камеры по окружности вокруг установки с помощью зажатой правой клавиши мыши в добавленном к ней скрипте прежде всего переменная типа Transform связывается с базовым объектом установки – точкой, вокруг которой будет вращаться камера с использованием метода RotateAroundtargetPos. Кроме того, отдаление и приближение камеры реализовано посредством вращения колесика мыши, а пределы перемещения камеры по сторонам проверяются отдельной функцией ControlDistance.

Лист. 2. Код управления камерой

```
public class Scroll : MonoBehaviour{
      [SerializeField]
      float scrollSpeed = 10f;
      [SerializeField]
     int sensivity = 3; int maxdistance = 20;
      int mindistance = 1;
      [SerializeField]
      Transform targetPos;
void FixedUpdate(){
     float x = Input.GetAxis("Horizontal");
      float y = Input.GetAxis("Vertical");
      if (x != 0 || y != 0) {
Vector3 newpos = transform.position + (transform.TransformDirection(new Vector3(x, 0, 0)) + Vector3.up * y) / sensivity;
  if(ControlDistance(Vector3.Distance(newpos, targetPos.position)))
```

```
transform.position = newpos; }
  if (Input.GetAxis("Mouse ScrollWheel") != 0) {
Vector3 newpos = transform.position + transform.TransformDirection(Vector3.forward * Input.GetAxis("Mouse ScrollWheel") * scrollSpeed);
  if (ControlDistance(Vector3.Distance(newpos, targetPos.position)))
transform.position = newpos;}
  if (Input.GetMouseButton(1)) {
transform.RotateAroundtargetPos.position,
Vector3.up, Input.GetAxis("Mouse X")*sensivity);
transform.Rotate(Vector3.left, Input.GetAxis
("Mouse Y")*sensivity); } }
  bool ControlDistance (float distance)
    {if (distance > mindistance && distance < maxdistance) return true; return false; }}
```

При разработке главного меню в шаблоне симулятора выделены два раздела: «УСТАНОВКА» и «ПРАКТИКА», выбор которых происходит после щелчка по соответствующей кнопке. Для того чтобы кнопки всегда находились в правом верхнем углу экрана вне зависимости от его разрешения, контейнеру UI, содержащему меню, добавлена соответствующая точка привязки к холсту Canvas. Данные кнопки имеют четыре разных состояния (покоя, наведения, нажатия и недоступности), в соответствии с которыми разработан их дизайн.

UI-интерфейс симулятора работает следующим образом:

- по умолчанию установка находится в рабочем состоянии и готова к выполнению эксперимента;
- не включая установку, можно осмотреть ее и окружение управляя камерой (клавишами клавиатуры для движения по осям координат и курсором мыши с нажатой правой клавишей для вращения камеры вокруг центра привязки установки);
- при наведении курсора мыши на пункт меню «УСТАНОВКА» выпадает вниз список всех элементов установки (рис. 2);
- при наведении курсора мыши на каждый элемент в списке подсвечивается его название, а также соответствующий элемент установки на сцене и в информационном окне отображается краткая информация об этом элементе;
- при щелчке мышью на элементе в списке камера плавно перейдет в положение для удобного просмотра выбранного элемента установки на сцене;
- при наведении на пункт меню «ПРАКТИКА» происходит исчезновение меню и симулятор переходит в режим начала работы (рис. 1).

Для программирования функциональности работы симулятора конкретной лабораторной установки на основе базового программного кода шаблонного симулятора придется заново запрограммировать взаимосвязи всех элементов создаваемого симулятора в соответствии с их предназначением в конкретной лабораторной установке. При использовании разработанного шаблона симулятора эта часть разработки нового симулятора является наиболее трудоемкой.

Прежде всего для каждого разрабатываемого симулятора лабораторной установки необходимо составить блок-схему взаимодействия всех ее функциональных объектов. При составлении такой блок-схемы нужно пройти следующие этапы:

1. Выделить те события, которые могут происходить (включение / выключение установки, поворот или перемещение какого-нибудь объекта, нажатие кнопки и т. д.).

2. Продумать, какие объекты должны инициировать эти события (например, нажатие по переключателю инициирует событие по включению / выключению установки).

3. Выбрать, какие элементы должны реагировать на то или иное событие (например, на включение и выключение установки может реагировать большое количество элементов, начиная от отображения показаний дисплеев до запуска механизмов).

4. Указать в блок-схеме, при каких условиях подписчики подписываются и отписываются от прослушивания требуемых событий и какие методы при этом будут вызываться.

Виртуальная лабораторная работа, выполняемая на симуляторе, содержит целый комплекс взаимосвязанных программных модулей. Для удобного и оптимизированного взаимодействия между модулями в шаблон симулятора внедрена система событий. Фактически система событий отражает архитектуру взаимодействия скриптов приложения и тесно связывается с построенной для каждого конкретного симулятора блок-схемой взаимодействия его объектов. Она представляет собой набор скриптов и правил, по которым в симуляторе происходит передача информации и взаимодействие между функциональными элементами установки. Система событий состоит из трех элементов: *Менеджер событий, Переключатель* и *Подписчик.*

Менеджер событий (лист. 3) содержит список всех событий, которые могут произойти во время работы с установкой.

Лист. 3. Фрагмент кода *Менеджера событий*

```
public class EventManager : MonoBehaviour {
public delegate void SwitchHandler(bool work); public delegate void MoveHandler();
//... делегаты для других событий
```

```
public static event SwitchHandler Switch; //создание события – переключение
//… создание других событий
private void Start()
      {     Switch(false);     }
   public static void SwitchWork(bool work) { Switch(work);     }
      public static void MovedCamera()
{     Moved();     }
//… методы вызова других событий
}
```

Переключатель (лист. 4) вызывает перечисленные в менеджере события.

Лист. 4. Код *Переключателя*

```
public class Switcher : MonoBehaviour {
      bool work = false;
//изначально установка не работает
private void OnMouseDown() {
//при нажатии мыши на переключатель
work = !work;
//меняется флаг на противоположный
   Power(work);
//вызывается метод инициировании события
   }
public void Power(bool work) {
      EventManager.SwitchWork(work);
//инициализация события
   }}
```

Экземпляр *Подписчика* (лист. 5) должен находиться на каждом элементе, который должен реагировать на событие.

Лист. 5. Код *Подписчика*

```
public class Follower : MonoBehaviour {
      new Renderer renderer;
      private void Start(){
      renderer = GetComponent<Renderer>();
      renderer.enabled = false;}
   void OnEnable(){
      EventManager.Switch += SwithWork; }
   private void OnDisable(){
      EventManager.Switch -= SwithWork; }
   void SwithWork(bool work{
      if(work) renderer.enabled = true;
else renderer.enabled = false; }}
```

В момент инициализации *Подписчика* он должен подписаться на требуемые события, а элементы на сцене должны содержать код вызова методов, находящихся в *Переключателе*. При вызове метода в *Переключателе* вызывается событие, соответствующее этому методу в *Менеджере событий*. При возникновении события подписанные на него подписчики обрабатывают соответствующую реакцию.

При разработке симулятора конкретной лабораторной установки необходимо, используя составленную для программирования работы симулятора блок-схему взаимодействия объектов, откорректировать в проекте Unity шаблонной установки код скриптов соответствующих классов системы событий: *EventManager* для *Менеджера событий* (лист. 3), *Switcher* для *Переключателя* (лист. 4), *Follower* для *Подписчика* (лист. 5).

Конечным этапом разработки симулятора является разработка последовательности заданий, выполняемых пользователем в ходе проведения эксперимента на виртуальной лабораторной установке. Последовательность выполняемых заданий фактически является ключевым модулем при работе с симулятором и опирается в свою очередь на использование заданной системы событий. В лист. 6 приведен фрагмент кода выполнения заданий в шаблонном симуляторе.

Лист. 6. Фрагмент кода выполнения заданий

```
public class Tasks : MonoBehaviour
private void Start(){
taskText.enabled = false;
taskText.text = ++numberOfTask + ". " + "Включите установку";
EventManager.Switch += WorkingControl;
EventManager.Switch+=Task1;
//подписываем первое задание на включение
}
Void WorkingControl(bool work)
// отслеживание, включена ли установка
      { isWorking = work; }
void Task1(bool w)
//первое задание, если включена установка
{if (w) {
 EventManager.Switch -= Task1;
//отписывание от выполненного задания
taskText.text = ++numberOfTask + ". " + "Поворотом барабана установите светодиод";
//вывод сообщения о новом задании
EventManager.ColorChanger += Task2;
//подписывание на второе задание и т. д.
}}
```

Каждое задание *Task1, Task2*, … сопровождается подпиской на конкретное событие и выводом соответствующего текстового сообщения. Первое задание для любого симулятора – это включить установку. Первое задание подписывается на событие включения установки и выводит в информационное окно текст: «Включите установку» (рис. 1). После нажатия пользователем мышью на включатель симулятора это приведет к выполнению первого задания. Затем программа отписывается от события включения для

экономии ресурсов. Далее программа откроет второе задание с подпиской на новое событие с соответствующим текстовым сообщением «Поворотом барабана установите светодиод». Таким образом, все задания будут последовательно выполнены.

Разработка 3D-моделей элементов для шаблонного симулятора лабораторной установки, а также его окружения в помещении выполнена в среде 3ds MAX, экспортирована в формате файла *.fbx, а затем импортирована в среду Unity для задания интерактивности функциональным объектам симулятора.

Для публикации разработанных симуляторов лабораторных установок из среды Unity на сайте используется программная библиотека WebGL – контекст элемента Canvas HTML, который обеспечивает API 3D-графики без использования плагинов. Макет сайта для доступа к теоретическим и практическим ресурсам виртуального лабораторного практикума разработан с использованием технологий HTML, CSS, Bootstrap и JavaScript.

Заключение. Предложенная технология и разработанный шаблон для симулятора лабораторной установки позволяют разрабатывать симуляторы лабораторных установок на основе заданного стандарта, что дает возможность существенно сократить время их разработки. Такие симуляторы полностью повторяют действия экспериментатора на реальной лабораторной установке благодаря запрограммированному функционалу их работы. Пользователь может получить все необходимые навыки работы с такими установками, чтобы затем применить их при работе в реальных условиях.

Таким образом, использование симуляторов лабораторных установок для выполнения практических заданий по различным техническим дисциплинам позволяет в принципе проводить такие практикумы индивидуально, вне лабораторных помещений и без присутствия преподавателя, что полностью соответствует целям обучения студентов в дистанционном режиме.

Список литературы

1. Гурин Н. И., Герман О. В. Компьютерные обучающие системы в издательском деле: в 2 ч. Минск: БГТУ, 2015. Ч. 1, 2. 372 с.

2. Гурин Н. И., Жук Я. А. Генератор семантической сети информационной системы в таблицу реляционной базы данных // Труды БГТУ. 2015. № 6: Физ.-мат. науки и информатика. С. 181–185.

References

1. Gurin N. I., German O. V. *Komp'yuternyye obuchayshchiye systemy v izdatel'skom dele* [Computer training systems in publishing]. Minsk, BGTU, Publ., 2015, part 1, 2. 372 p.

2. Gurin N. I., Zhuk Ya. A. The information system semantic network generator to a relational database table generator. *Trudy BGTU* [Proceedings of BSTU], 2015, no. 6, Physics and Mathematics. Informatics, pp. 181–185 (In Russian).

Информация об авторах

Гурин Николай Иванович − кандидат физико-математических наук, доцент кафедры информационных систем и технологий. Белорусский государственный технологический университет (220006, г. Минск, ул. Свердлова, 13а, Республика Беларусь). E-mail: ngourine@mail.ru

Сахонь Елизавета Сергеевна – магистрант кафедры информационных систем и технологий. Белорусский государственный технологический университет (220006, г. Минск, ул. Свердлова, 13а, Республика Беларусь). E-mail: elizavetakotik@gmail.com

Information about the authors

Gurin Nikolay Ivanovich − PhD (Physics and Mathematics), Assistant Professor, the Department of Information Systems and Technologies. Belarusian State Technological University (13a, Sverdlova str., 220006, Minsk, Republic of Belarus). E-mail: ngourine@mail.ru

Sahon Elisaveta Sergeevna – Master's degree student, the Department of Information Systems and Technologies. Belarusian State Technological University (13a, Sverdlova str., 220006, Minsk, Republic of Belarus). E-mail: elizavetakotik@gmail.com

Поступила после доработки 12.01.2021