Список цитируемых источников

1. Microsoft Visual Studio [Электронный ресурс].— Режим доступа : https://www.tadviser.ru/index.php/Microsoft_Visual_Studio/. — Дата доступа : 11.04.2021.
2. SQLite [Электронный ресурс]. — Режим доступа : https://ru.bmstu.wiki/SQLite. — Дата доступа : 11.04.2021.
3. *Шапович, Е. Г.* Системы сокрытия и шифрования информации с использованием стеганографии / Е. Г. Шапович, А. В. Шах // Сборник трудов IV Международной научно-практической конференции, 2018. — С. 101—104.

УДК 004.94

М. А. Кононович
Учреждение образования «Барановичский государственный университет», Барановичи, Республика Беларусь

ОТНОШЕНИЯ МЕЖДУ КЛАССАМИ И ОБЪЕКТАМИ НА ПРИМЕРЕ ИГРОВОГО ПРИЛОЖЕНИЯ

Введение. Компьютерные технологии развиваются с огромной скоростью. В современном мире компьютеры помогают в работе, используются в сфере образования, экономики, игр, досуга, позволяют людям легко связываться друг с другом.

Объектно-ориентированное программирование — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования [1].

Существует много разных типов отношений, которые два объекта могут иметь в реальной жизни. Для описания этих отношений используются определенные слова для их описания, например, является, имеет, использует, зависит, является частью и т.д. Все эти типы отношений имеют свои аналоги в виде отношений в объектно-ориентированном программировании.

Основная часть. Между классами возможны два типа отношений:

1. Отношение типа is-a (есть, является), при котором один класс есть подвидом другого класса. При таком отношении один класс расширяет (детализирует) возможности другого класса. Расширение возможностей класса осуществляется благодаря использованию наследования.
2. Отношение, при котором существует взаимосвязь между двумя классами. Здесь выделяют два подвида взаимосвязи между классами:
   - отношение типа has-a (класс содержит другой класс). В этом случае в классе объявляется одних или несколько экземпляров другого класса. При данном отношении возможны два случая взаимодействия. Первый случай, это когда объект (экземпляр), который объявлен в классе, не является составной частью класса (агрегация) и его использование не влияет на функциональную работу класса. Второй случай, когда объект, который объявлен в классе, есть составной частью этого класса (композиция);
   - отношение типа uses (класс «использует другой класс). В этом случае класс содержит программный код другого вложенного класса, к которому он имеет доступ [2].

В качестве объекта исследования выбраны отношения между классами и объектами в объектно-ориентированном программировании. В качестве предмета исследования выбран игровой движок Unreal Engine 4. Целью исследования выступает анализ отношений между классами в реализованной игре.

Unreal Engine 4 — это игровой движок и редактор, разработанный компанией Epic Games для создания игр и приложений от мультиплатформенных игр с миллионными бюджетами до мобильных инди-разработок. Unreal Engine работает на операционных системах Windows и macOs и разработанные в нем проекты могут быть развернуты на платформах Windows, Mac, PlayStation 4, Xbox One, iOS, Androidm, HTML5 и Linux. В самом простом формате Unreal Engine 4 является коллекцией редакторов, используемых в различных стадиях производства игр или приложений [3].

Для анализа отношений между классами была создана игра графического изображения аквариума на языке программирования C++. Приложение позволяет управлять персонажем в виртуальный 3D мире аквариум. Пользователь может перемещаться в пространстве и рассматривать виртуальный 3D мир.

При создании проекта были созданы и использованы следующие основные классы:

1. UObject — базовый класс, от которого наследуется подавляющее большинство объектов.
2. Landscape — класс, позволяющий создать ландшафт. При надобности изменять его размеры, добавить углубления и возвышенности и т. д. (рисунок 1).
3. Map — класс для генерации объектов пространства. Генерация объектов пространства представлена на примере создания и расположения объекта «сундук» (рисунок 2).
4. Light — класс, предназначенный для просчёта освещения на карте.
5. Character — класс, позволяющий создать персонажа, установить определенные его настройки, а также прописать логику его действий (рисунок 3).

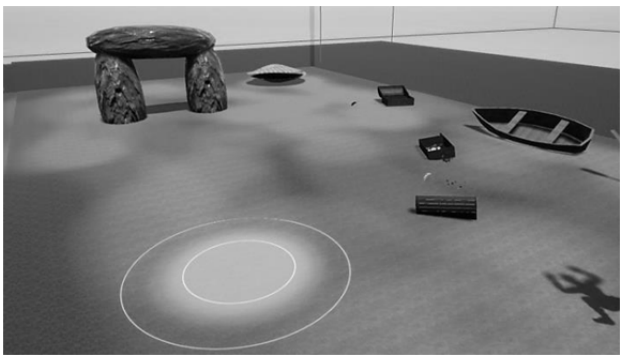

Рисунок 1 — Создание ландшафта

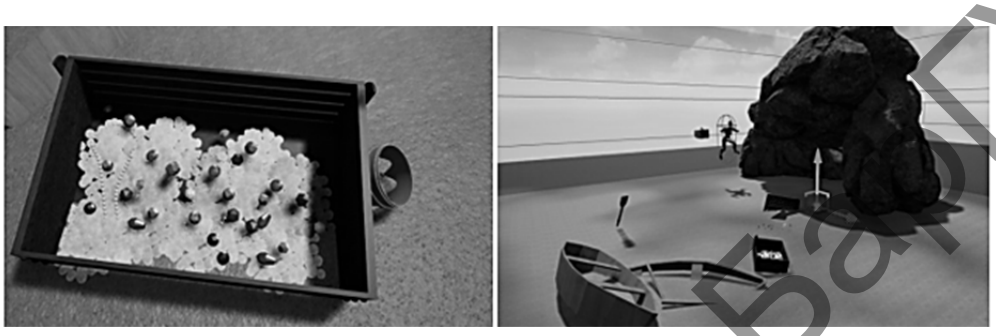

Рисунок 2 — Расположение сундука с сокровищами на карте

Рисунок 3 — Созданный персонаж

6. PhysicsVolume — класс для задания области, в которой персонаж может плавать.

Диаграмма классов созданного игрового приложения представлена на рисунке 4.

Используемые отношения между классами:

1. Наследование представляет один из ключевых аспектов объектно-ориентированного программирования, который позволяет наследовать функциональность одного класса или базового класса в другом — производном классе. На примере данного исследования от класса UObject наследуется объекты ,которые создаются в мире или просто в памяти: это персонаж, ландшафт и объекты пространства.

2. Композиция служит для выделения специальной формы отношения «часть-целое», при которой составляющие части в некотором смысле находятся внутри целого. Таким образом «аквариум» и персонаж являются частью пространства.

3. Ассоциация — это отношение, при котором объекты одного типа неким образом связаны с объектами другого типа. На данном примере пространство использует источники освещенности.

4. Агрегация — это отношение при котором объект одного класса имеет в своем составе объект другого класса, что продемонстрированно на примере классов ландшафт и пространство. То есть пространство имеет в своем составе ландшафт.

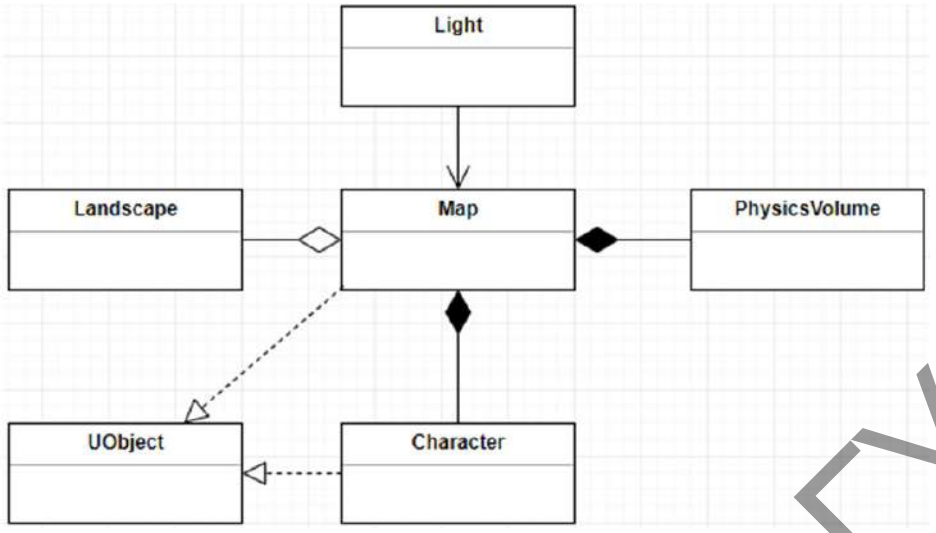

Рисунок 4 — Диаграмма классов приложения

Заключение. В ходе данного исследования были получены и усвоены важные и полезные навыки работы в игровом движке Unreal Engine 4, а также использования отношений между классами и объектами.

После акцентирования внимания на классах и объектах при проектировании и разработке игровых приложений на ранних стадиях внимание программиста сосредоточивается на внешних проявлениях ключевых абстракций и механизмов. Такой подход создает логический каркас системы: структуры классов и объектов. На последующих фазах проекта, включая реализацию, внимание переключается на внутреннее поведение ключевых абстракций и механизмов, а также их физическое представление. Таким образом, формируется архитектура системы, которая включает архитектуру процессов, и архитектуру модулей.

Список цитируемых источников

1. *Павловская, Т. А.* C/C++. Программирование на языке высокого уровня / Т. А. Павловская. — СПб. : Питер, 2005 — 461 с.
2. Паттерны объектно-ориентированного программирования / Э. Гамма [и др.]. — СПб. : Питер, 2020. — 448 с.
3. *Куксон, А.* Разработка игр на Unreal Engine 4 за 24 часа / А. Куксон, Р. Даулингсока, К. Крамплер. — Москва : Эксмо, 2019. — 528 с.

УДК 004.9

А. А. Косак, **В. А. Полубок**
Институт информационных технологий учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники», Минск, Республика Беларусь

РАЗВИТИЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ С ИСПОЛЬЗОВАНИЕМ ОНЛАЙН-СИМУЛЯТОРОВ

Введение. На протяжении более десяти лет дистанционное обучение используется учреждениями образования и по праву заняло свое место среди других форм образования. А в настоящее время, в связи с эпидемиологической ситуацией, оно превратилось в насущную необходимость. Но даже когда наступят те благословенные дни, когда преподаватели и студенты полностью вернутся в аудитории, без сомнения, дистанционное обучение все же останется. Это говорит о том, что становление дистанционного обучения как одной из форм образования состоялось.

Основная часть. Дистанционный формат обучения, как и любой другой формат, это эксперимент, и требует гибкого подхода. Необходимо определить, какое время уйдет на изменение и отладку учебного процесса, придется тестировать различные сценарии и инструменты. Важно не пытаться повторить офлайн-обучение, а найти удобный вам и вашим студентам подход к дистанционному формату.

Посмотрим, как в большинстве случаев устроено обучение. Студент просматривает вебинары и потом выполняет задания. Этот процесс повторяется несколько раз. В конце — тестирование.

Основная проблема такого подхода — пассивное восприятие. По ходу вебинара нет аудиторного общения, нет вопросов лектора, нет небольших заданий для лучшего усвоения материала. Студенты пассивно