Содержание

Предисловие

Многие, кто приходит к программированию в Roblox уже прошли школу Scratch. Это естественно, ведь 3D-мир гораздо интереснее, чем мир плоский. Да и разрешение картинок в Scratch гораздо хуже, чем в Roblox. И это — не говоря о многих недоступных в Scratch спецэффектах и возможностях!

Но, как говорится, лучшее имеет свою цену! Программирование в Roblox Studio сложнее, чем программирование в Scratch. Книг нет, и первое, что приходит на ум — поискать в Youtube, ведь “там есть все”! Но, этот подход к созданию игр имеет свои “минусы”. Конечно же, вы сможете посмотреть ролики в Youtube, скопировать приведенные в них коды и вставить их в свою игру. Открытыми останутся лишь вопросы: “А как же работает код, который вы скачали? Почему этот код устроен именно так?” На эти вопросы многочисленные видеоблогеры ответов не дают: ”Копируй! Вставляй! И все будет работать!” Но, так программистами не становятся, и настоящей серьезной игры вы таким способом не создадите. Это однозначно!

Если вы это понимаете — эта книжка для вас. Одним из ее соавторов является такой же, как вы школьник 13-ти лет. Четыре года назад он тоже начинал в Roblox с поиска подсказок в Youtube, но в 10 лет его папа сказал: “Читай туториал!” И он начал почитывать. А это — трудно, ведь туториалы Roblox написаны на английском. Нужно пользоваться программой перевода с английского. А если вам примерно столько же лет, как и ему, вам не нужно объяснять, что многие термины даже в переводе на русский могут быть абсолютно непонятны! Но, он был упорен и много экспериментировал. Без проб и ошибок в программировании — никуда! Сегодня у моего соавтора много своих Roblox-игр, есть и дипломы конкурсов по программированию. Получал он поощрения и донаты от самого Roblox. Но, он не расслабляется — продолжает упорно “грызть гранит науки”. Roblox Studio и язык программирования Lua — вещи сложные, а “слона нужно есть по частям”!

Наша книга содержит описание инструментов и решений, которые позволят вам не только разработать свои собственные Roblox-игры, но и найти ответы на многие вопросы, на которые не отвечают виде-

облогеры. Мы очень старались, чтобы все, что описано в этой книге было вам понятно.

И наш совет: ищите друзей, которым тоже интересна разработка компьютерных игр, советуйтесь с ними, делитесь с ними полезной информацией и, обязательно, привлекайте их к разработке совместных проектов. Программирование — не для одиночек! Каждому найдется работа по интересам. Кто-то станет дизайнером вашей игры, кто-то разработает классный сценарий, а кто-то наполнит игру крутыми спецэффектами! Желаем вам успехов!

Полезные подсказки:

* при копировании текстов программ из книги **никогда** не ставьте никакие знаки препинания в конце строк кода (даже если они есть в книге);
* набирайте тексты программ **только в английской раскладке** клавиатуры. Каждый символ (даже визуально очень похожие символы кавычек **“ ”**) в русской и английской раскладках имеют разные коды и если вы вместо “английских” используете “русские кавычки”, Roblox воспримет это, как ошибку, а вы не сразу ее обнаружите;
* **регулярно сохраняйте свою работу**, например, через каждые 10 минут. Таким образом, если что-то пойдет не так, по крайней мере, вы не потеряете многого. При сбоях ПК используйте копии ваших программ, которые Roblox Studio автоматически сохраняет в папке **Roblox - AutoSave**;
* не редактируйте программы в то время, как они запущены — ваши изменения не будут действовать — нужно остановить программу, исправить скрипт и запустить программу заново;
* при разработке и отладке программ **всегда** держите открытыми окна **Output** и **Script Analysis,** чтобы контролировать сообщения об ошибках, которые Roblox Studio выводит в эти окна. Эти сообщения выделяются красным цветом шрифта — пытайтесь читать, что написано, в окне **Script Analysis** и **Output**;

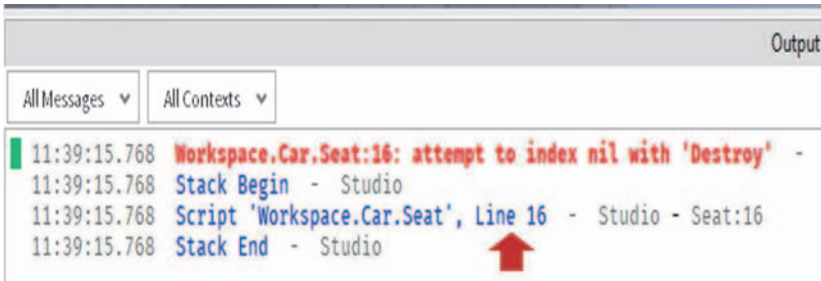

Рис. 0.1 Пример сообщения об ошибке программы в окне Output

В приведенном на рисунке примере показано сообщение об ошибке программы при попытке исполнения строки 16 (**Line 16**) скрипта с названием **Seat**, входящего в состав объекта **Car**, являющегося составной частью **Workspace**;

* при отладке программы не держите открытой программу, которую не отлаживаете. Окно **Output** будет работать только в одной из открытых программ и, вполне возможно, не в той, что нужно;
* для отладки программ регулярно пользуйтесь функцией **print()**, распечатывая те или иные переменная, чье значение должно меняться в ходе программы и, по изменению чьей величины, можно судить о том, правильно ли исполнилась какая-то команда или нет;
* когда создаете **parts** в меню **Part** Roblox Studio, а затем копируете в них заимствованный где-либо или даже в этой книге скрипт, всегда ставьте в скрипт реальные координаты **parts** на вашем игровом поле, а не координаты, которые указаны в заимствованном скрипте, иначе скрипт может не работать.

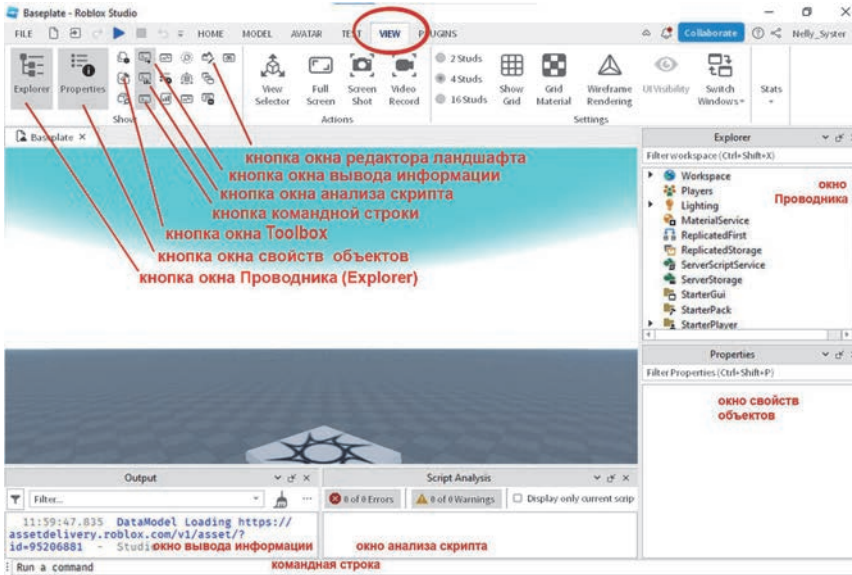

Рис. 2.4 Основные окна Roblox Studio и кнопки их вызова

Заметим, что когда вы вызываете какое-то из окон, оно может появиться в случайном месте экрана, что часто бывает неудобно.

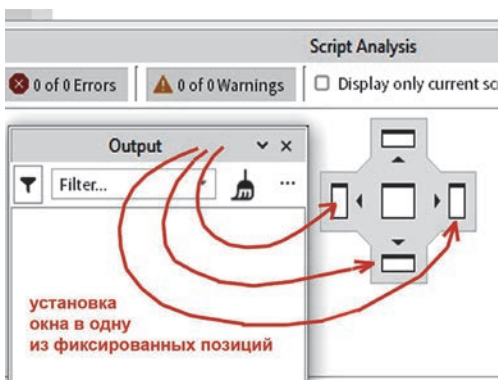

Рис. 2.5 Пример перемещения одного из окон (окна Output) для закрепления его в удобной позиции

Поставьте курсор мыши на “шапку” появившегося окна, нажмите левую кнопку мыши и чуть сдвиньте окно — появится окно с “геймпадом”. Выбрав одно из его положений, вы можете закрепить выделенное окно в той фиксированной позиции, где вам будет удобно.

В окне Проводника (**Explorer**) всякий раз показан полный исходный состав объектов создаваемой вами игры. Левее некоторых перечисленных в **Explorer** объектов стоят направленные вправо стрелки. Если кликнуть по такой стрелке, она развернется острием вниз, а под объектом, рядом с которым она стоит откроется список объектов, вхо-

— свойство **Holdduration** определяет, сколько секунд необходимо удерживать кнопку, прежде чем ее нажатие приведет к нужному результату. Если указать значение “0”, сработка управления произойдет сразу же;

— свойство **ClickablePrompt** указывает, может ли пользователь кликнуть и удерживать подсказку мышкой, чтобы запустить управление событием (можно выбрать **true** или **false**). Если же пользователь использует телефон или планшет, он всегда может взаимодействовать с подсказкой близости, щелкая по ней независимо от свойства **ClickablePrompt**.

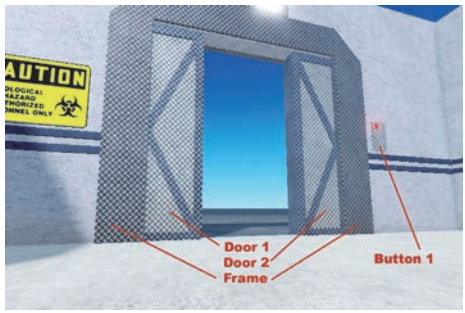

Рис. 10.5 Пример применения *Proximity Prompt* в конструкции раздвижных дверей

Приведем пример использования **Proximity Prompt**. Создайте что-то подобное тому, что изображено на рис. 10.5. Как это выглядит в окне **Explorer** показано на рис. 10.7.

Рис. 10.6. Вид подсказки о близости, включающейся при подходе игрока к кнопке Button1, управляющей открыванием/закрыванием раздвижных дверей (a) и индикатор времени удержания кнопки “E” перед началом открывания/закрывания дверей (b)

```
63              if not success then
64                  warn(errorMessage)
65              end
66      end
67      local function onPlayerRemoving(player)
68              local playerID = player.UserId
69              if playerGold [playerID] then
70                  saveData(playerID, playerGold [playerID])
71              end
72      end
73      Players.PlayerAdded:Connect(onPlayerJoin)
74      Players.PlayerRemoving:Connect(onPlayerRemoving)
```

16. Инструменты и стреляющее оружие

Рис. 16.1 Создание инструмента

Инструменты — это то, что игрок может держать в руке и использовать в игре. Это может быть оружие или, даже, продукты питания. Инструменты можно хранить в игровом мире или раздавать игрокам. В этой главе вы узнаете, как создать лазерный бластер.

Любой инструмент в Roblox исходно называется **Tool**. Давайте создадим его:

— вставьте новый объект в **Workspace**. Для этого кликните по кнопке ⊕ правее **Workspace** и выберите в выпадающем меню **Tool** — появится объект инструмент с таким же названием;

— измените его название на **Blaster**. Кликните по кнопке ⊕ правее **Blaster**, выберите в меню объект **Meshpart** и вставьте его в **Blaster**. Кратко поясним: **Meshpart** — это 3D-модель, разработанная в другой специальной программе, например, в **Unity**, но найти которую можно на сайте Roblox. В нашем случае, мы используем 3D-модель бластера;