Иерархическая структура объектов Godot

Язык GDScript используется для написания скриптов в игровом движке Godot

Скрипт в Godot не может существовать сам по себе. Он выполняется только в том случае если он прикреплён к какому-либо объекту

Объекты в Godot также называются нодами или узлами

У каждого узла в Godot есть родительская надо или сцена, за исключением корневого узла (root). Таким образом объекты образуют иерархическую структуру, как в файловой системе

Группы объектов образуют сцены и заготовки. Фактически заготовки — это тоже сцены, но как сцены они не используются. Заготовки используются для включения в другие сцены, в этом случае они становиться обычными узлами, но при этом сохраняют всю группу своих объектов

Любая сцена представляет собой иерархию. Каждая сцена имеет корневую ноду.

Каждая нода имеет свой путь: root/название сцены/название ноды. С его помощью можно обращаться к любому объекту на сцене.

Помимо пути, также ноды имеют имя, при этом объекты могут называться одинаково, но при условии, что они не являются дочерними для одного и того же узла. Или же, важно, что бы пути до объектов не совпадали

Можно привести аналогию с файловой системой. Сцена – это папка, а узел – это файл. При этом узлы сами могут выступать в роли объекта

Также существуют глобальные скрипты – синглоты. Они запускаться вместе с запуском игры. Для синглота не требуется объект.

Также при запуске игры, первой запускается главная сцена. Она указывается в настройках проекта. Сцены можно переключать, внутри запущенной сцены можно создавать новые объекты. При переключении сцены, старая сцена удаляется, доступ к ней утрачивается и загружается новая сцена. При повторной загрузке сцены, она загрузится по новой

Синтаксис GDscript

Синтаксически GDscript похож на Python, но многие аспекты взяты из Java Script

Godot имеет некоторые области видимости, в которых можно или нельзя писать определённые команды

Самой верхней областью, является область объявления. В ней можно объявлять глобальные переменные, функции и классы, но нельзя писать команды

Команды и подпрограммы можно прописывать внутри функций, так же внутри функций можно объявлять локальные переменные, которые существуют только в контексте этой функции

Отделение областей видимости выполняется с помощью табулирования, как и в python

Комментарии в GDScript начинаться со знака решётки “#”

Встроенные функции объекта

Каждый объект имеет встроенные функции, которые можно переопределить. По умолчанию они ничего не выполняться, хотя и вызваться всегда

\_init – выполняется при инициализации объекта

\_ready – когда объект (нода) готова и существует на сцене

\_process – выполняться один раз за кадр

\_physics\_process – выполняется 60 раз в секунду

Переменные, типы данных

В GDScript переменные объявляться с помощью двух ключевых слов. Это ‘var’ и ‘const’.

В отличии от других языков программирования, в GDScript ‘const’, обозначает не константу, а модификатор доступа. ‘var’ означает, что к переменной можно обратиться из другого объекта, а ‘const’ скрывает переменную от других объектов.

Т.е. константы в Godot не являются постоянными. В рамках объекта, константы ведут себя также, как и переменные. Можно привести аналогию: ‘var’ – это ‘public’, a ‘const’ – ‘private’

Далее, я буду говорить только о ‘var’, но всё тоже самое относиться и к ‘const’

После ключевого слова ‘var’, указывается название переменной. На данном этапе переменная уже объявлена. Она может принимать значения любых типов.

При объявлении можно сразу же задать значение переменной. Для этого после названия переменной пишется знак присваивания (‘=’) и значение которые мы хотим присвоить.

При этом тип данных переменной зависит от самих данных. И при перезаписи данных, тип может измениться. При каких-либо операция, типы данных также могут неявно приводиться к общему типу, если это возможно.

Для того чтобы явно указать тип данных, нужно после названия переменной, но перед знаком присваивания, написать двоеточие, а после тип данных

В Godot существуют следующие, встроенные, простые типы данных:

null – это пустой тип данных, не содержащий информации и не имеющий никакого другого значения.

bull - тип данных может содержать только истину (true) или ложь (false).

int - целочисленный тип данных может содержать только целые числа (как отрицательные, так и положительные).

float - используется для хранения значения с плавающей запятой (действительные числа).

string - последовательность символов в формате Unicode.

Также, есть, встроенные, сложные типы данных:

Vector2, Rect2, Vector3, Matrix32, Plane, Quat, AABB, Matrix3, Transform, Color, Image, NodePath, RID, Object, InputEvent, Array, Dictionary

Привидение переменных

Для приведения данных к определённому типу, нужно использовать специальные функции. Эти функции не изменяют тип данных у переменной. Они возвращают новое значение.

Так, например функция int(). Если применить её к типу данных bool, то вернуться 0 или 1, если к string, то вернуться все числа из строки, как число.

Математические операции:

Математические операции идентичны операторам в математике и в других языках

Для сложения используется оператор ‘+’

Для вычитания используется оператор ‘-’

Для умножения используется оператор ‘\*’

Для деления используется оператор ‘/’

Для получения остатка используется оператор ‘%’

Особенности математической операции сложения:

Математический оператор сложения позволяет складывать числа и проводит конкатенацию строк. Если значения числового типа, то произойдёт сложение чисел. Если строкового – то выполниться конкатенация строк. Иначе выброситься ошибка ‘Invalid operand for operator’

Логические операции

Для Godot идентично использование таких операторов сравнение как меньше ‘<’, больше ‘>’, не больше ‘<=’, не меньше ‘>=’, равно ‘==’, не равно ‘!=’

Для конъюнкции используется оператор ‘and’ или ‘&&’

Для дизъюнкция используется оператор ‘or’ или ‘||’

Для импликации используется оператор ‘not’ или ‘!’

Результатом логический вычисления является, true или false

Функция приведения к логическому типу - bool(). Возвращает false, если первым параметром передать 0 (для int), 0.0 (float) или пустую строку;

Массивы

Массивы, описываться на GDScript двумя квадратными скобками. Массивы могут включать в себя, разные типы данных. Отсчёт индексов, ведётся с нуля т.е. первый элемент имеет индекс 0

Массивы, как и все переменные в GDScript являются ссылками.

Для обращения к элементам массива, нужно указать индекс элемента в квадратных скобках, сразу после названия переменной

Массивы имеют следующие методы:

size() вернёт длину массива. Длина массива – это кол-во элементов в нём.

append() добавит элементы в конец массива

Словари

Словари, описываться на GDScript двумя фигурными скобками Словари могут включать в себя, разные типы данных. Индексами в словарях могут быть любые символы. В качестве индексов могут использовать переменные. Тогда индекс, станет равным значению переменной

Для обращения к элементам словари, нужно указать ключ элемента в квадратных скобках, сразу после названия переменной.

Также можно обращаться к элементу, через точку

Словари имеют следующие методы:

has() вернёт true, если словарь содержит ключ, который был передан в параметры

values() вернёт все значения в виде массива

keys () вернёт все значения в виде массива

остальные функции описанны здесь ссылка

Ветвления в GDScript

Оператор if проверяет выражение после себя. Если выражение возвращает true, то выполняется блок кода после if . Если условие не верно, то выполняется оператор elif. Он ведёт себя также как и оператор if, но этот оператор не обязателен. Операторов elif может быть сколько угодно. Если все операторы elif получили false, то выполняется оператор else.

Оператор match проверяет соответствие переменной какому-либо значению. Каждое значение описывается отдельным блоком. C помощью знака нижнего подчёркивания ‘\_’, можно задать блок, который выполнится в случае, если переменная не соответствует ни одному значению.

Тринарный оператор в GDscript имеет следующий вид: 1 if bool = true else 0  
Данное выражение вернёт единицу, если переменная bool равняется true, иначе false