Prof. Monster's

BGE Guide to

The Game Loop

By Monster
Version 1.0
30 Dec 2011

Hi Blenderheads

If you are reading this, you are most likely thinking of making a game. The BGE is a great platform to do that.

This guide should help you to understand how the BGE and finally your game works.

There is one thing that every game has:

THE GAME LOOP

I do not want to detail what a game loop is. I'm pretty sure you can find a lot of information in wikipedia or if you google for it.

Nevertheless I want to present you how the Game Loop is build up in the Blender Game Engine (BGE).

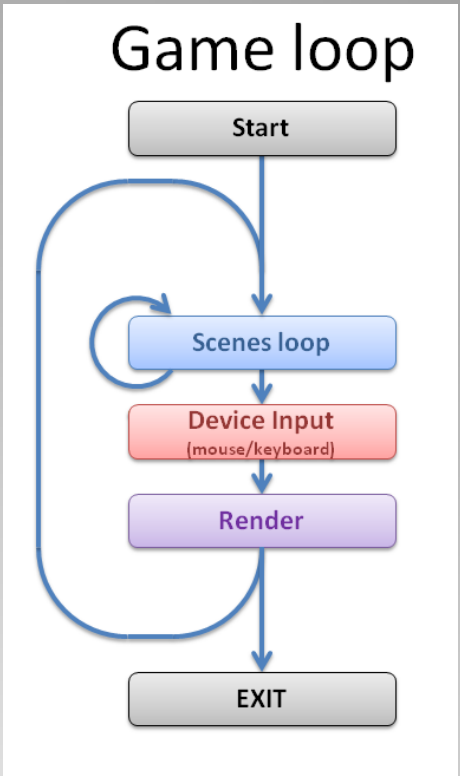
It is essential to know how it works. If you keep that in mind it will explain a lot of "mysterious" behavior that you might have discovered in the past.

I hope you enjoy reading.
Monster

Purpose

Each cycle in the Game Loop represents one logical frame. Also known as logic tick. It is the smallest time unit within your game.

1. The scenes are processed
2. The devices are checked for input
3. The final image is rendered.

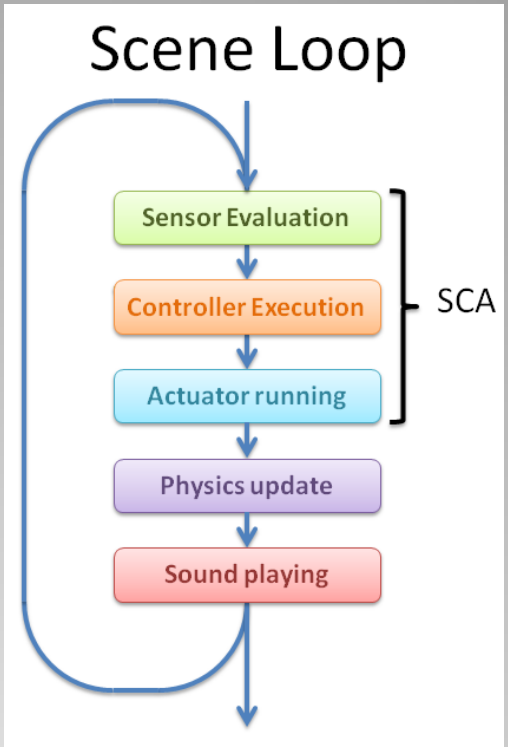


It is important to know, that the render part might be skipped if the last frame was eating to much processing time (lag).
There is a limit how much renders can be skipped. If this limit exceeds (default = 5), a render will take place regardless how long it takes. Such "render" lags will result in "logic" lags. This let the game run slower than expected.

The Game Loop

This loop is a bit more complex.

The scenes loop cycles through all active scenes performing following steps for each scenes:



1. Sensor evaluation
2. Controller execution
3. Actuator running
4. Physics update
5. Sound playing

1..3 builds the SCA concept that you configure as Logic bricks. The logic bricks are the interface to the logic of your game.

This is the order. Keep it in your mind!

- You will never see a controller executed before a sensor is evaluated.
- You will never see an actuator running before all controllers are executed.
- The physics update will be done after the actuators were running but before the render is drawn!

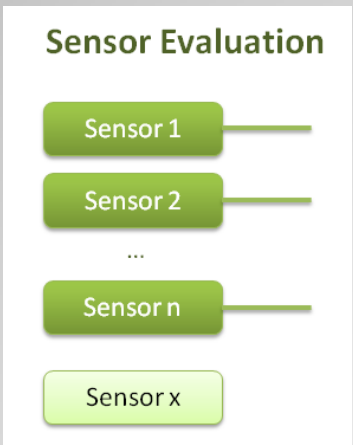
The Scene Loop

Active sensors are ALWAYS evaluated regardless what parameters you set. There is no parameter that prevents a sensor to be evaluated at every frame.

Sensors are active if they are connected to a controller of an active state.

This means if a sensor is not connected to a controller of an active state it gets not evaluated. This becomes quite handy if you want to stop the evaluation of "heavy" sensors like radar or ray.

Dependent on their parameters evaluated sensors trigger their connected controllers



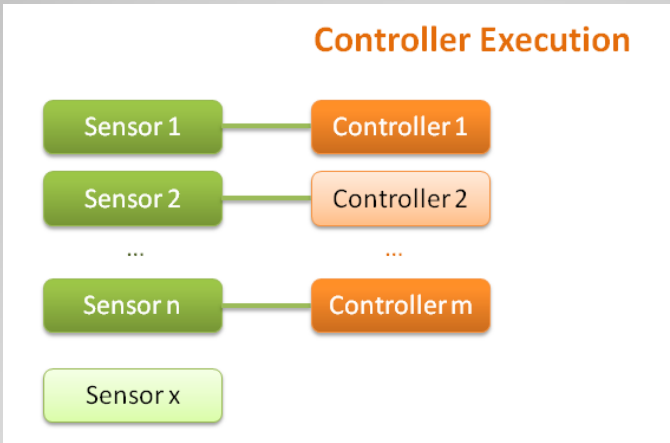
SCA -Sensors

All triggered controllers are executed, regardless if the sensor's state is True or False.

ONLY triggered controllers are executed, regardless if the sensor's state is True or False.

There is no way to execute controllers that are not triggered by their connected sensors. Therefore it is very important to choose the right sensors for your controllers.

Executed controllers send activation or deactivation signals to their connected actuators dependent on the Sensor input.



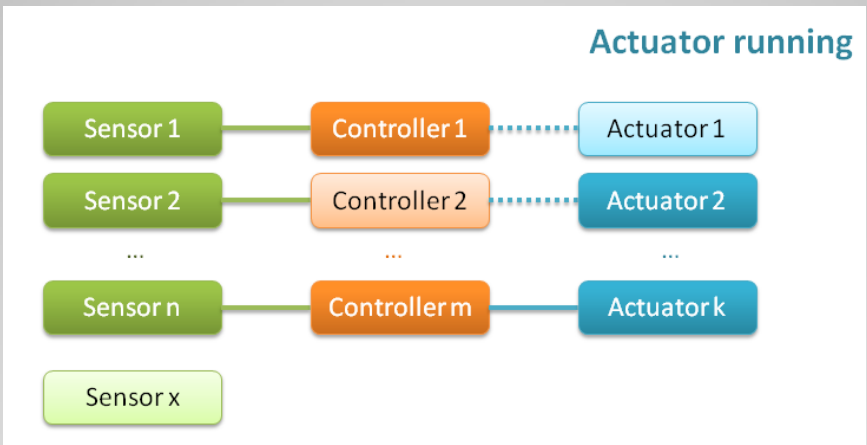
SCA -Controllers

Activation and deactivation is input from a connected controller. For that reason we call them activation signal and deactivation signal.

Receiving these signals does not necessarily mean the actuator gets activated or deactivated. What it does depends on the Actuator's implementation and parameters.

How long an actuator stays active depends on the actuator type and parameters.

There are actuators that deactivate after one frame, others wait for deactivation signals, or stay active until they are done ignoring the deactivation signal.



SCA -Actuators

The SCA concept builds a quite efficient event system. It allows to separate events from resulting tasks. Performing resource intensive tasks only when necessary.

The Blender GUI allows an easy set up of the logic.

By the nature of the GUI the logic is limited on True/False sensor state and Activation/Deactivation controller signals.

To increase flexibility there is the option to use Python as Logic brick. For whatever reasons it is available as controller only.

This allows you to create your own customized controller.

Beside that the Python controller can perform Actuator's tasks. That means it can change the game's current state. That is something that only actuators can do.

If you look back into the scenes loop you see that controllers are executed before actuators run. That means that Python code is executed before the actuators run. This is a small but important fact.

Python in SCA

Please note:
Some changes to the game are not processed immediately (e.g. ending an object, switching a scene). They are placed in a queue and processed later at an more appropriate time. This can sometime lead to confusion.

Performance:
The Python code should exit as fast as possible.

- Long code runs cause game lags.
- Endless loops cause a game freeze.

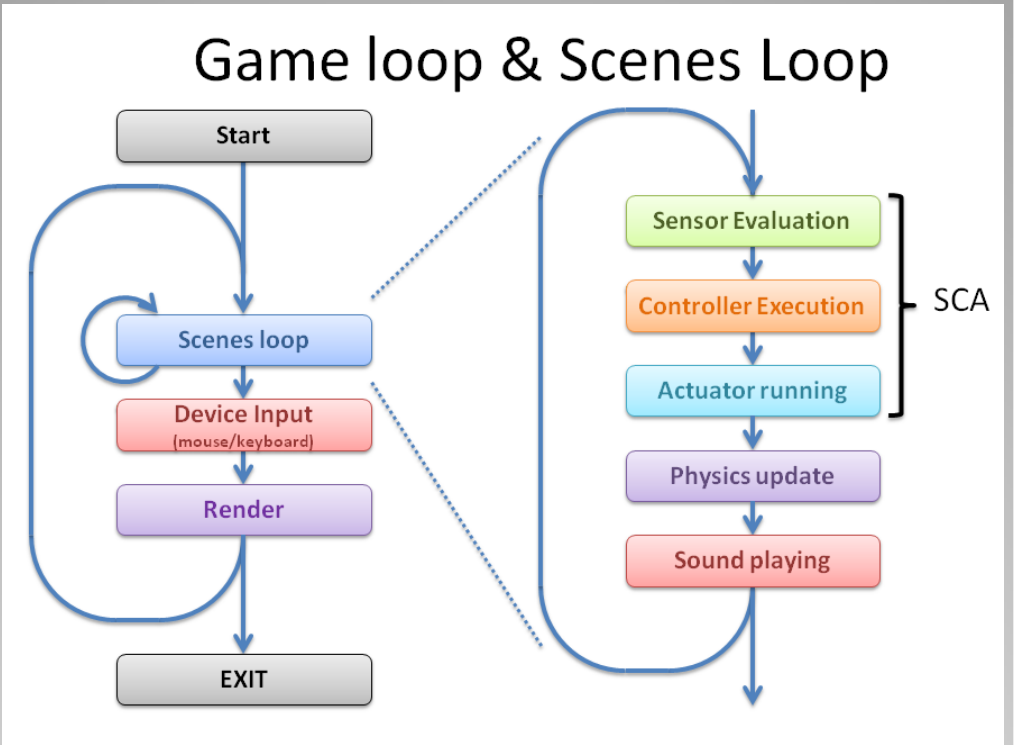
It is faster not to execute a controller rather than executing a controller that does nothing. For that reason it is important to choose the right sensor configuration. Avoid to unnecessarily trigger connected controllers. Keep in mind the Python controller is the “heaviest” controller in terms of processing time.

Logic bricks are usually faster then Python code. Therefore it is not recommended to recreate existing logic bricks with Python (e.g. AND controller, motion actuator). It is usually better to change the configuration of a logic bricks especially if the parameters do not change frequently.

Python can be more efficient if one piece of code processes the purpose of multiple logic bricks in one step (e.g. validating keyboard input, adding multiple objects).

Python hints

Finally both loops in one picture:



You might miss details on Render. It is excluded here as the details of the rendering are not important to the logic of the game.

Keep in mind:

- Shaders do not influence logic or physics
- Logic influences Shaders.

Overview

You read how the Game Loop runs in the BGE. This should provide you with an idea what runs when in your Game.

I hope it helps you to create a fancy game without discovering to much obstacles on your way.

The road to success is long and I hope this little information can point you into the right direction.

You can find me under www.blenderartists.org as Monster

Good luck
Monster

Thank you