FUNCTIONALITY OF PYTHON PROGRAMMING LANGUAGE IN BLENDER'S GAME ENGINE

Starkova A., student,
Storchak O. H. Associate Professor,
Kharkiv National University of Radioelectronics

Abstract. This paper deals with the functionality of Python in the Blender's game engine. Python is integrated with Blender to extend functionality, automate tasks and create a custom toolkit. The Python programming language offers Blender numerous advantages for interactive game development to gain a unique gaming experience. Python is used to create objects, animations, add-ons, logic for game objects, custom operators, to model and render objects, to interact with game objects and to work with external libraries.

Key words: Python, Blender, functionality, game development.

This investigation aims to explore the use of the Python programming language in the Blender game engine. The subject of the investigation is the functionality of Python in the Blender's game engine. The investigation is of topical interest as it concerns the growing popularity of Python in game development and the increasing use of Blender as a game engine. The integration of Python with Blender's game engine offers more possibilities for game developers. Research materials are scholarly and technical literature on the topic of the Python programming language and the Blender software. The general scientific methods of the research are analysis, synthesis and comparison.

The Python programming language and the Blender software are studied by researchers Victor Kuller Bacone [1], Conlan Chris [2], Felinto Dalai and Pan Mike [3], Jan van den Hemel [4], Brian Kent [5], Mark Lutz and David Ascher [6] to name a few.

Blender is 3D creation software that is often used for developing animations, games, visual effects and 3D models. The Blender's native file format is a binary type. Binary files are written in such a way that a user can't get to them directly. These files are designed to be accessed by programs and not by users.

The Blender suite can run the Python programming language. Python is commonly applied for image processing with Blender that sets up its Python environment and embeds a Python interpreter which is started with Blender and stays active. This interpreter runs scripts to draw the user interface and it is used for some of Blender's internal tools. Scripts are plain text files. A script can be opened in any text editor and immediately see the differences between two similar files. Blender provides the bpy module to the Python interpreter. The bpy module is the core of the Blender Python API (Application Programming Interface). This module can be imported in a script and give access to Blender data, classes and functions. Scripts that deal with Blender data need to import this module. Almost every user interaction is tied to a Python function. Since Python can be used with modules written in other languages, it can be used with other applications. The Blender Python API allows integration for: bpy.types.Panel, bpy.types.Menu, bpy.types.Operator, bpy.types.PropertyGroup, bpy.types.KeyingSet and bpy.types.RenderEngine. In game development, Python can be used to interact with game objects, create game logic and automate tasks. Blender defines methods for Python integration. This works by creating a Python subclass of a Blender class which contains variables and functions specified by the parent class. Python is more focused on producing readable code. In the Python code, your own classes, modules, and elements can be created.

The easiest way to integrate the Blender game with the exterior world is with Python [5]. Almost everything that you can do with Python, you can do in the game engine. Many Python scripts come bundled with Blender and can be used as a reference because they use the same API that script authors write tools in.

The Blender's game engine is the part of the Blender 3D editor used to create actual 3D video games [1]. It's the ideal entry level game development environment because you don't even need to learn to program. The game engine itself is written in C++, an object-oriented language, and Python OOP capabilities let us handle the game data in a Python-native way. Much like C++, Python supports both procedural and object-oriented programming modes. The game engine API is a bridge connecting Python scripts with game data. It reflects in the game engine objects having their own

set of functions and variables directly accessed from a Python API. Python variables are very flexible: they can be declared on the fly when you first use them; you can assign different data types for the same variable and name them dynamically. In the game engine, the script interface is controller-centric by design. Therefore, the Python script can be considered as a complex controller.

On startup Blender scans the 'scripts/startup/' directory for Python modules and imports them. The exact location of this directory depends on your installation. Here are some ways to run scripts directly in Blender: load in the text editor and press *Run Script*, type or paste into the interactive console, execute a python file from the command line with Blender.

Python scripts can be edited in the Blender's text editor that has some special properties pertaining to imports, system paths and linked files in Python. Python scripts can access objects by their names and write directly to their data blocks. Running Python scripts in the text editor is useful for testing. It is possible to extend Blender to make tools accessible like other built-in functionality.

Scripts can be run as modules. The obvious way is the 'import some_module' command from the text window or interactive console. If you open it as a text block and tick 'Register' option, this will load with the blend file. If you copy it into one of the directories 'scripts/startup', they will be automatically imported on startup. If it is defined as an add-on, it will enable the add-on to load it as a Python module. The only difference between add-ons and built-in Python modules is that add-ons must contain a *'bl_info'* variable which blender uses to read metadata such as name, author, category and URL.

One of the main features of using Python in Blender's game engine is its ability to interact with game objects. This feature allows developers to create complex game mechanics and automate tasks in the game development process [1].

Another feature of Python in Blender's game engine is its ability to create logic for game objects. This is achieved through the use of logic bricks, which are visual elements that represent logical operations. For instance, a Python script can be used to detect collisions between objects and trigger specific actions, such as playing a sound

effect or displaying a message on the screen. The colour of an object changes when it collides with another object. This can be achieved by creating a Python script that is attached to the object. The script contains code that listens to collision events and changes the colour of the object when a collision occurs. Python can be used to create complex game mechanics such as pathfinding for AI-controlled characters or real-time physics simulation for objects in the game world. [3].

The next feature of using Python in the Blender's game engine is its ability to work with external libraries. Python external libraries can grab data off the Internet for game score, control your game with a controller, combine Head-tracking and immersive displays for augmented reality. The Python library must be compatible with the Python version that comes with your Blender. Python code can invoke C and C++ libraries, can be called from C and C++ programs [6, p. 4]. Python has a vast ecosystem of libraries and modules that can be used to extend its functionality. By importing these libraries into Blender, users can access their features and use them in their game development [2, p. 121–126]. The Pygame library can be used to create a simple 2D game within Blender's game engine. Pygame provides a set of tools for creating games, including support for graphics, input and audio. By importing Pygame into the Blender's game engine, users can take advantage of its features and create more complex games [2, p. 219–222].

Custom operators can be created using Python in Blender [2]. Operators are actions such as moving an object or applying a modifier in Blender. By creating custom operators, users can automate complex or repetitive tasks, making their workflow more efficient. To create a custom operator, the properties of the operator, such as what it will do and how it will be triggered, must first be defined. This is done using Python classes and decorators, which provide a way to define the behavior and appearance of the operator. Once the operator is defined, it can be registered with Blender's user interface. This allows users to trigger the operator from within the Blender interface by pressing a button or selecting a menu item. Custom operators can also be assigned to hotkeys, making it easier to trigger them.

The Python programming language in Blender can also be used to create objects, animation and add-ons, to modify the properties and parameters, to model objects and to render. Python can create meshes, lights, cameras, realistic scenes, render high-quality images, adjust rendering settings to create videos, plug-ins to import and export files, automation tools, additional toolbars and then modify their shapes, materials and textures. Python can create motion paths, control the speed of objects or change animation properties. The Python API in Blender allows programmers to create complex animations that can be automated or customized for different effects, to customize rendering settings such as lighting, shadows, rendering settings for cameras and views, to create add-ons that extend Blender's functionality and add new tools and features.

Thus, the integration of Python with Blender increases functionality and provides numerous advantages for game developers to create interactive games. The integration allows developers to automate tasks, extend functionality, to create custom toolkit, objects, animations, add-ons, logic for game objects, custom operators, to model and render objects, to interact with game objects and to work with external libraries.

References

1. Bacone Victor Kuller. Blender Game Engine: Beginner's Guide. 2012. 206 p.
2. Conlan Chris. The Blender Python API: Precision 3D Modeling and Add-on Development. 1-st ed. 2017. 158 p. URL : https://books.google.com.ua.
3. Felinto Dalai, Pan Mike. Game Development with Blender. 2013. 448 p.
4. Hemel Jan van den. Blender Secrets. Vol. 1. 2019. 328 p.
5. Kent Brian R. 3D scientific visualization with Blender. 2015. 92 p. URL : https://github.com/mikepan/GameEngineBook.
6. Lutz Mark, Ascher David. Learning Python. 4th ed. 2009. 1161 p. URL : https://cfm.ehu.es/ricardo/docs/python/Learning_Python.pdf.