Scene staging environment

Helder Daniel

Departamento de Engenharia Electrónica e Informática
Faculdade de Ciências e Tecnologia
Universidade do Algarve

September 25, 2021

Abstract

This paper presents the Scene Staging Environment (SSE) game engine. This game engine was developed with two aims: a simple 2D game engine for Python built on PyGame 2 (Shinners [2011, 2020a]) and an environment to develop and test artificial intelligence (AI) algorithms.
The environment is presented as a theater stage, where entities known as Actors perform on a Stage following a Role script. The Stage itself is the universe, that can have a animated background scenario and can be fully or partially captured by a Camera, that presents its view at the user interface (UI). Together with TensorFlow (Abadi et al. [2015]) machine learning package, SSE provides an environment suitable to develop and test diverse AI algorithms.

1 Introduction

1.1 SSE game engine

The Scene Staging Environment (SSE) extends Pygame 2, to provide a simple 2D game engine. Pygame is defined as "a set of Python modules designed for writing video games." (Shinners [2011, 2020b]). Pygame itself is built on top of the Simple direct Layer (SDL) graphics library. SDL is a cross-platform library, built with C language, to achieve fast performance. It gives "low level access to audio, keyboard, mouse, joystick, and graphics hardware via OpenGL and Direct3D" (org [1998, 2021])). It has bindings to several programming languages such as C++ and Python.

Pygame has low level support for developing games, such as 2D graphic sprite, collision detection and sound support, giving support to develop "fully featured games and multimedia programs". However this low level features makes the development unnecessary verbose, when compared with Higher Level game engines like: Unity (technologies [2014, 2021], Haas [2014]) and Unreal (Games [2019, 2021]).
This higher level game engine, provide support for defining entities or actors, interacting with an environment, making the game development process simpler, faster and the source code cleaner and also simpler.
SSE is not aimed at AAA game development, however provides an higher layer on top of Pygame to manage Actors interacting with each other and with an environment at an higher level, simplifying and speeding up the development of games.
There are other higher level game engines projects available for Python are Pygame zero (zero org [2021], Pope [2021]), (arcade org [2017, 2021], Craven [2017]) Arcade and Panda3D (org [2002, 2021]) also available for C/C++.

1.2 SSE AI bench

Since the dawn of computer gaming, AI algorithms have been used to give non-player characters (NPC) some believable behavior. War game simulations like Eastern Front, (at the army unit level) and Balance of power (at the global conflict level) are such early examples form the 1980's (Crawford [1984]) Balance of Power (Crawford [2003])
Since then the algorithms used to control NPCs have never stopped being improved, wether designed for war gaming or any other kind of game. Maybe due to their still popularity, simplicity and also some nostalgy, there are currently some projects aimed at providing an environment to run AI algorithms and test them with retro games (Surma [2018a], Surma, Surma [2018b], AI [2016]) SSE ability to control NPCs with controller classes that follow a role or script, makes it suitable to be bound with AI platforms such as TensorFlow.

2 Operation overview

SSE implements a theater metaphor where a game is ran on a Stage with a background and animated Scenario and players are Actors playing roles 1. Actually, besides the Scenario, everything that moves is an Actor, including explosions and other visual effects.
A Camera takes pictures of the full, or parts of the stage with a desired frame rate. Moving the Camera on a large stage it is possible to show context action. For instance the camera can be locked on an Actor showing the surroundings.

Figure 1: SSE Stage, Scenario, Actors and Camera

Currently SSE supports only one camera, but it is planned to add support for several cameras, so that different views of the Stage can be shown on the user interface (UI).

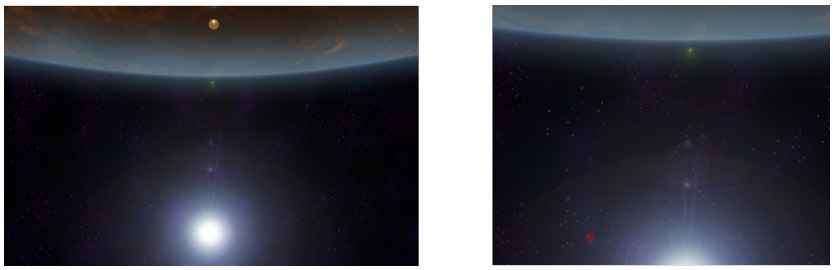

Figure 2: Full stage capture and zoom

Figure 3 presents the main classes UML diagram. Class Game represents instances of a game. A Game has one or more stages. A different stage can implement a different level. A stage can have one or more background scenarios. The scenarios can be animated to give the illusion of player movement, for instance ships across a space background. The Stage has also a Camera.
Actors belong to a Stage. The behavior of Actors depend on Class Role. Role can automatize the behavior of an NPC character implementing an suitable algorithm. A sub class of Role is the Controller. This class allows human player to control character using a device such as a game controller, the mouse or keyboard.
Actor and scenario animation is implemented with FlipBook class. This class

holds a set of images that can be changed between frames to give the illusion of animation.

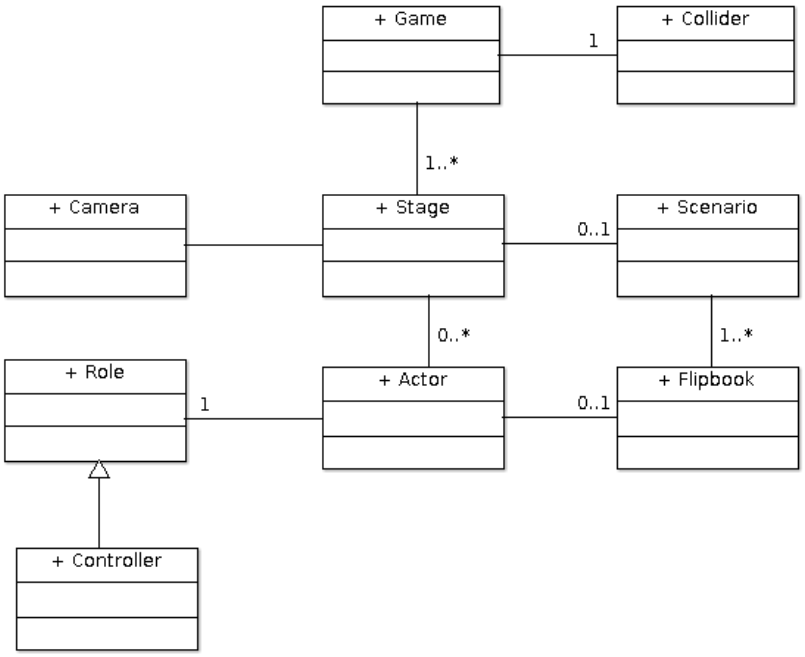

Figure 3: SSE main classes

Game class has also an instance of the Collider class. This class is responsible to manage collisions at Actor type level. Each Actor object has a type used to choose the action when 2 collide: a missile colliding with a ship results in the destruction of both a the creation of another Actor, an explosion.
To manage visual effects, such as explosions, Actor class has a time to live. After that time expired it is removed.

The action is performed by placing actors in the Stage. Each actor as a Role to follow. Weapons such as missiles are also Actors placed on staged when fired. When they hit something the Collider manages the collision. The game engine manages all the action. Actors have only to be placed at a time with a given Role, in a fire an forget fashion. The SSE game engine will manage all the interaction.

3 Case study: sooth'em up

With SSE implementation of 2D games as Shoot'Em ups are quite straight forward. Below is presented the implementation of an Invaders like game.

The Scenario can be painted with stars and other celestial bodies and be animated to scroll giving the illusion that all the ships are traveling across space.

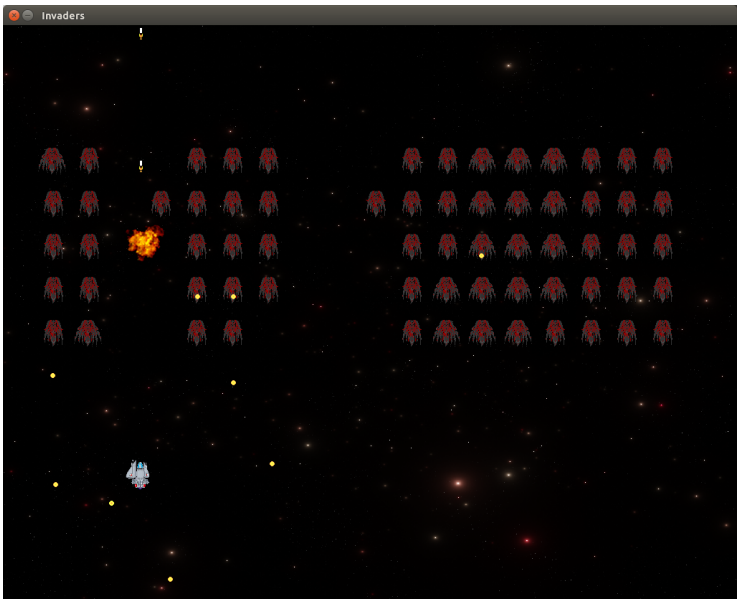

Figure 4: SSE Invaders like shoot'em up game

Aliens are placed with a Role, that can be as simple as semi random move towards the human Player ships, or with more intelligent behavior, to elude Player missiles.
Ships can be animated with FlipBooks to give more realistic movement. Explosion and other effects such as shields can have a time to live to be removed from the stage after a given time. The Collider will handle their interaction.

4 Conclusion

The Scene Staging Environment current version, supports the development of 2D games, such as the shoot'em up addressed in this text.
The ability to control (Non) Player Charaters with scripting roles, provides support for any control algorithm, including machine learning algorithms developed

with TensorFlow.
In the near future the SSE will serve as an environment to develop and rehearse reinforcement learning algorithms.

References

Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems. 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Open AI. Gym, a toolkit for developing and comparing reinforcement learning algorithms, 2016. URL https://gym.openai.com/. [Online; accessed 24-August-2021].

Python arcade org. The python arcade library, 2017, 2021. URL https://api.arcade.academy/en/latest/. [Online; accessed 28-November-2021].

Paul Vicent Craven. Arcade documentation, 2017. URL https://readthedocs.org/projects/arcteaching/downloads/pdf/latest/. [Online; accessed 28-November-2021].

Chris Crawford. The art of computer game design, 1984.

Chris Crawford. On game design, 2003.

Epic Games. Unreal engine, 2019, 2021. URL https://www.unrealengine.com. [Online; accessed 11-December-2021].

J. K. Haas. A history of the unity game engine, 2014.

Lib SDL org. Sdl 2.0, 1998, 2021. URL https://www.libsdl.org/. [Online; accessed 28-November-2021].

Panda 3D org. Panda 3d, 2002, 2021. URL https://www.panda3d.org/. [Online; accessed 28-November-2021].

Daniel Pope. Pygame zero documentation 2.1, 2021. URL https://readthedocs.org/projects/arcteaching/downloads/pdf/latest/. [Online; accessed 28-December-2021].

Pete Shinners. Pygame 2.0, 2011, 2020a. URL https://www.pygame.org/. [Online; accessed 28-November-2021].

Pete Shinners. Pygame 2.0 wiki, 2011, 2020b. URL https://www.pygame.org/wiki/about. [Online; accessed 28-November-2021].

Greg Surma. Atari - solving games with ai (part 2: Neuroevolution), year = 2018, url = https://gsurma.medium.com/atari-reinforcement-learning-in-depth-part-1-ddqn-ceaa762a546f, note = "[online; accessed 24-august-2021]",.

Greg Surma. Atari - solving games with ai (part 1: Reinforcement learning), 2018a. URL https://gsurma.medium.com/atari-reinforcement-learning-in-depth-part-1-ddqn-ceaa762a546f. [Online; accessed 24-August-2021].

Greg Surma. Ai research environment for the atari 2600 games, 2018b. URL https://github.com/gsurma/atari. [Online; accessed 24-August-2021].

Unity technologies. Unity game engine, 2014, 2021. URL https://unity.com. [Online; accessed 11-December-2021].

Pygame zero org. Pygame zero, 2021. URL https://pygame-zero.readthedocs.io/en/stable/. [Online; accessed 28-December-2021].