Reinforcement Learning in Game Engines

Sri Ganesh, Bharat Kotwani, Ajay Ganesh Suryawanshi , Kamal Mandava, Abhay Bansod, Ushasree Tella, Lanka Sai Krishna Chaitanya, Sujay Bhagawan Ghadge, Gaurav Kakoti, Deepak Kari

Abstract

Machine Learning has developed in many ways during the recent years. One of the new fields which has potential applications in machine learning is computer games. With this research I aim to develop machine learning functionalities in computer games using a lightweight 2D - Game Engine developed in Java. The engine will have the basic functionalities of any Game Engine and we will be able to use it to create simple neural networks and train these neural networks for use in the game.
The neural network will be trained using data collected from the game and we have expectations that the model will even be able to learn how to play the game by itself.
For the development of the Game Engine we will be using tools and libraries for Java for example, LWJGL 3, OpenGL, Assimp, Maven for Research Management and Research Structure Management, GLFW for event listeners, GLSL for Shaders, using ImGUI for Graphical user interface development. Serialization and Deserialization with GSON to save our designed levels. Gradle is used for Dependency and Library Management, Launch4j for converting the Packaged .jar file into an executable for Distribution.

1. INTRODUCTION

1. THEORETICAL BACKGROUND

A Game Engine is a software or tool that assists the user in creating video games. A game engine consists of several key concepts and methods which are essential for an optimized use of underlying hardware and drivers which process and determine how exactly the Graphics are Rendered and presented on screen to the user.
The modern graphics pipeline discarded the use of the earlier pure Raster Graphics method and incorporated several steps to gain more control over the data. This multi-step pipeline involves the use of Vertex Shaders, Shape Assembly, Geometry Shader, Rasterization, Fragment Shader and Blending to generate much more clear realistic and colorful images.

1. MOTIVATION

As video games are becoming more and more popular among generations, both young and old, the demand for interactiveness and use of modern technologies to enhance the gaming experience has increased. Machine Learning as a field has developed a lot in recent years and is already becoming a part of the gaming industry in various ways. Some of the current uses of Machine Learning in games has been through NPC(Non- Player Characters) control and PCG(Procedurally Generated Content). Organizations like Valve have also incorporated Machine Learning to improve the performance of bot's in their games like CSGO.
However, as most of the gaming industry has been focusing on languages such as C++, C and C# as the primary tools for creation, other languages like Java still lack a lot of the tools for such uses. I have taken up this research as a way to fill some of this void in the marketplace and offer alternatives to other similar libraries or frameworks.

1. AIM

My aim with this research is to create a working 2D Game Engine from scratch using Java and add integrated ML functionalities to it. This will allow other people who might use my developed engine to create games, to use

Machine Learning tools directly from the game engine. I also aim to create an easily accessible and usable Graphical User Interface to simply the usage of the Game Engine for the layman. The game engine will contain most of the basic features in other similar 2D Game Engines and allow us to expand over that by allowing us to incorporate ML in the developed games.

1. OBJECTIVES

The objective with this research is to develop a method to integrate Machine Learning tools and methodologies in video games based in Java using a self-made 2D game engine to incorporate Data Collection and Reinforcement Learning. Then Packaging these tools together so that they can even be used by laymen easily for their own games they design using the game engine. This will also help in enriching the java library ecosystem and offer alternatives to other people with similar needs.

2. Literature Survey

Name	Citation	Salient Features	Year	Link	Observation
Game Engine Architecture and Comparative Study of Different Game Engines	C. Vohera, H. Chheda, D. Chouhan, A. Desai and V. Jain, "Game Engine Architecture and Comparative Study of Different Game Engines," 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2021,	Comprehensive study and comparison of the most popular Game Engines for both Open-Source and Licensed.	2021	https://ieeexplore.ieee.org/document/9579618	The latest iteration of Unreal Engine (UE5), is much more optimized for modern computers and is easier to work with. Unity is the best choice for people who want to develop games on a comparatively smaller scale without loss of performance.

	pp. 1-6, doi: 10.1109/ICCCNT51525.2021.9579618.				
3D Game Development using LWJGL 3	–	The tools used are all open source. Has an in depth guide about Backend Logic in Game Engines.	2022	https://lwjglgamedev.gitbooks.io/3d-game-development-with-lwjgl/content/	The logic and Mathematics behind the graphics, rendering, and view models are systematically explained.
Alan Gavain: Blogs and Computer Concepts		Has a concise explanation of the basics and programming patterns of Game Engines	2018-23	https://alain.xyz/blog	Explains the various complicated System Designs and Features pertaining to Graphics, Game Engines and its components

Exploring the Use of Machine Learning as Game Mechanic – Demonstrative Learning Multiplayer Game	J. Dornig and C. Li, "Exploring the Use of Machine Learning as Game Mechanic – Demonstrative Learning Multiplayer Game Prototype," 2020 IEEE	Explores innovative uses of AI to make games more interesting and adaptive to user behavior instead of pre-scripted content. Explores uses	2020	https://ieeexplore.ieee.org/document/9175245	Details

Prototype	Conference on Multimedia Information Processing and Retrieval (MIPR), Shenzhen, China, 2020, pp. 396-399, doi: 10.1109/MIPR49039.2020.00087.	of game resources to collect data for ML.			
Applied Machine Learning for Games: A Graduate School Course	Applied Machine Learning for Games: A Graduate School Course Yilei Zeng, Aayush Shah, Jameson Thai, Michael Zyda University of Southern California {yilei.zeng, aayushsh, jamesont, zyda}@usc.edu	Paper on Procedural Generation of Levels using Neural Networks.	2021	https://ojs.aaai.org/index.php/AAAI/article/view/17849/17654	Contains content on Human-AI Interactions and Simulated Interactive Environments
OpenAI GYM	TY - JOUR AU - Brockman, Greg AU - Cheung, Vicki AU - Pettersson, Ludwig AU -	This is a large community based tool which contains pre-built environments and libraries that can be used to test,	2016	https://www.researchgate.net/publication/303822031_OpenAI_Gym	It is interactive and easy to learn and use. Can be used to test and compare results from your model.

	Schneider, Jonas AU - Schulman, John AU - Tang, Jie AU - Zaremba, Wojciech PY - 2016/06/05 SP - T1 - OpenAI Gym ER -	build and compare your Machine Learning Models.			
Mario AI Model using Pytorch and OpenAI		Easy to learn and follow tutorial on building machine learning models for games. Use of popular and interactive tools to simplify the AI model.	23/12/21	https://www.youtube.com/watch?v=2eeYqJ0uBKE	Use of OpenAI and NES simplifies the process of creating a wrapper environment for the game, which enables us to use inputs and collect data from the game.
Stable-Baselines3: Reliable Reinforcement Learning Implementations	Stable-Baselines3: Reliable Reinforcement Learning Implementations Submitted 12/20; Revised 10/21;	Study on use of metrics to compare machine learning models using stable baselines. Provides a simple API to allow easy	2021	https://www.jmlr.org/papers/volume22/20-1364/20-1364.pdf	StableBaselines can be integrated with Pytorch and can accurately and justifiably record metrics such as losses and episodic returns.

	Published 11/21 Antonin Raffin1 Ashley Hill2 Adam Gleave3 Anssi Kanervisto4 Maximilian Ernestus5 Noah Dormann1 Journal of Machine	access, well documented, high quality implementations, open-source and community supported.			
	Learning Research 22 (2021) 1-8				

3. Overview of Proposed System

To design the system I have used a combination of open-source tools and custom libraries which support graphics, rendering and the other related aspects of the task. The brief description and uses of these tools has been given:-

1. LWJGL: LWJGL or Lightweight Java Game Library which is used for development of games and graphics for Java. It also has support for audio and is a go to choice for including graphics in a java research.

2. OpenGL/GLFW: OpenGL is a cross-platform, cross-language library/API that provides functionalities for Vector Graphics and is used along with toolkits like GLFW to handle Windowing and hardware.

3. Assimp: Open Asset Importer Library is used to load assets(sprites, effects, animations) into one unified data structure.

4. Maven: Apache Maven is a software research management and comprehension tool. Based on the concept of a research object model (POM), Maven can manage a research's build, reporting and documentation from a central piece of information.

5. Gradle: Gradle manages and automates library and dependency management for researchs. It has an ecosystem of plugins which can be used for the integrating and handling processes.

6. JBOX2D: JBOX2D is a close port of a C++ Physics Library called BOX2D. It also has the same extensive feature and provides functionality for Rigid Body Dynamics and Collision Detection and Resolution.

7. ImGUI: It is a bloat-free minimal GUI builder, which can be used along with GLFW and LWJGL3 to design and create a GUI for our game engine.

8. DeepLearning4j: It is one the only Java Libraries which supports deep learning in Java and has extensive compatibility features with Python. We can use Python to train the model and then import the Tensorflow or Pytorch Models to use directly using DeepLearning4j.

9. JOML: JOML or the Java OpenGL Math Library is a java based Math Library which is used to perform complex linear algebra calculations needed by any Game Engine for Rendering and Graphical purposes.

10. Launch4j: Launch4j is a cross-platform tool for wrapping Java applications distributed as jars in lightweight Windows native executables. The executable can be configured to search for a certain JRE version or use a bundled one, and it's possible to set runtime options, like the initial/max heap size. The wrapper also provides better user experience through an application icon, a native pre-JRE splash screen, and a Java download page in case the appropriate JRE cannot be found.

Using these tools we can design the basic framework of research which consists of the following main modules:-

1. Rendering, Shading, Meshing and Batching
2. Animation System
3. Game Engine Physics
4. User Interface
5. Data Collection and Logging
6. Machine Learning Library
7. Evaluation Metrics and Scoring
8. Packaging
9. Distribution

Description of Modules

1. **Rendering**: Rendering is the process involved in the generation of a two- dimensional or three-dimensional image from a model by means of application programs. Rendering is mostly used in architectural designs, video games, and animated movies, simulators, TV special effects and design visualization. The techniques and features used vary according to the research. Rendering helps increase efficiency and reduce cost in design.

2. **Shading**: Shading is referred to as the implementation of the illumination model at the pixel points or polygon surfaces of the graphics objects.

Shading model is used to compute the intensities and colors to display the surface. The shading model has two primary ingredients: properties of the surface and properties of the illumination falling on it. The principal surface property is its reflectance, which determines how much of the incident light is reflected. If a surface has different reflectance for the light of different wavelengths, it will appear to be colored.

3. **Meshing:** The meaning of meshing–or mesh generation–is: defining continuous geometric shapes (such as 3D models) using 1D, 2D, and 3D shapes (mesh faces). The finer the mesh, the more accurately the 3D model will be defined.
Though meshes can be created manually, most meshing today is performed using

software, with minimal human input.

4. **Batching:** Every game engine needs to generate data using the Central Processing Unit (CPU) on your motherboard, and then transfer this data over to the Graphics Processing Unit (GPU) on your video card so that it can render things to the screen. When rendering different data objects, it is best to organize the data in groups so that you minimize the number of calls from the CPU to the GPU. You also want to minimize the number of state changes which can kill your game's performance. The group that holds the data to be rendered is called a batch.

5. **Animation System:** Although there can be multiple animation systems depending upon need, in this research we have a simple Finite State Automata Design for the animations. That is, the various animations trigger only based on the previous state and action of the entities involved.

6. **Game Engine Physics**: Computer animation physics or game physics are laws of physics as they are defined within a simulation or video game, and the programming logic used to implement these laws. Game physics vary greatly in their degree of similarity to real-world physics. Sometimes, the physics of a game may be designed to mimic the physics of the real world as accurately as is feasible, in order to appear realistic to the player or observer. In other cases, games may intentionally deviate from actual physics for gameplay purposes. Setting the values of physical parameters, such as the amount of gravity present, is also a part of defining the game physics of a particular game.

7. **User-Interface**: The user interface is a very important part of determining the viability and usability of any software. Regardless of how useful or necessary a software is, it doesn't matter if the layman is unable to use the software to its fullest capabilities without being versed in the relevant fields. The User-Interface built for the Game Engine is easy to grasp and use intuitively even for people with little to no experience in Computer Science.

8. **Data-Collection and Logging**: During the development of a game or any software, there is a need for feedback from software regarding errors and exceptions and also in our case, collecting game-based data to use in developing Machine Learning functionalities.

9. **Machine Learning**: The Game Engine also includes a custom library for creating and training Neural Networks as well using these neural networks in games, in real- time or otherwise, using saved models.

10. **Evaluation Metrics:** Video games need various indicators to determine how well a player is doing. The game engine engine has support for querying the game state and keeping track of score, which can also be customized by the developer.

11. **Packaging**: The game engine needs to be packaged before distribution as either an installer or executable single file including the required dependencies libraries. This was achieved for our purposes by using JAR's and a library called Launch4j which converts JAR's into a single Executable file format.

System Architecture:-

The system architecture consists of mainly 3 parts which interact with each other in a loop called the Game Loop. The game loop contains all the information and events of the game and is the core of the software design.

When starting the application the layer which triggers first is called the Scene Layer. A scene is basically a distinct state of the program that holds information and some way of getting from one scene to the next. The first scene layer initializes the game loop and adds context for the creation of the Window and its specific properties and attributes. It also contains different event triggers which perform specific functions when those events occur.

The game model is stored in memory as a long data type address of the first memory location and this model is then rendered or displayed on the screen to the user after a series of processes. Once the application has started and is running on the game loop, the client can use the GUI to interact and place items on the screen depending on how they want to construct their game. The user can also use several features in the game engine to import their custom assets and program entities to interact a certain way.

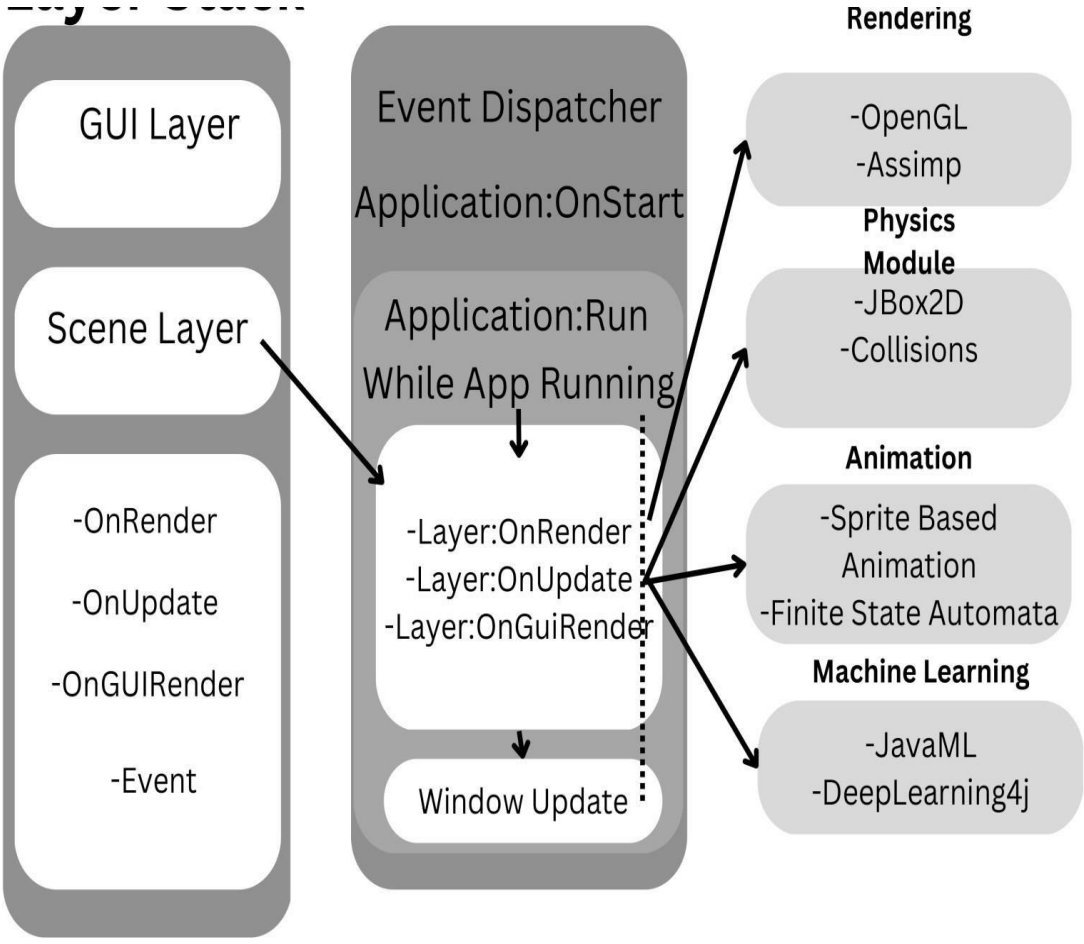


fig. 1.1 Overview of System Architecture

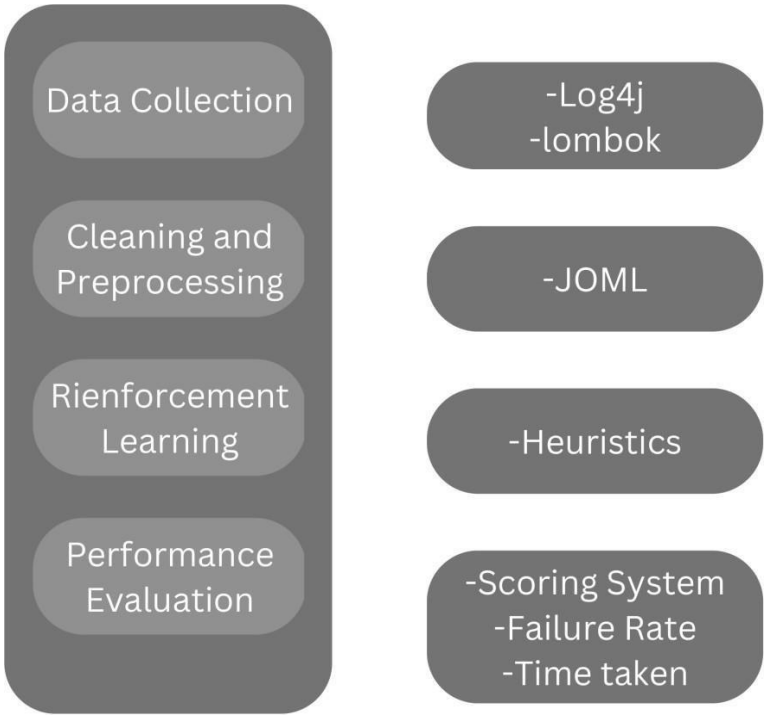


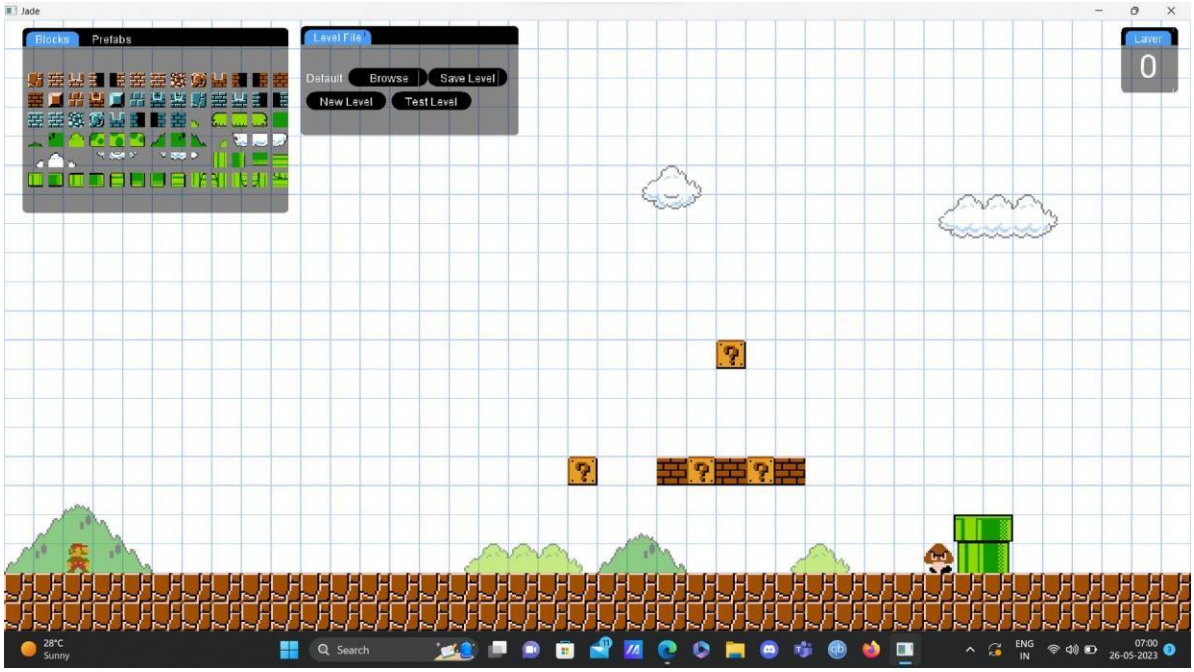
fig. 1.2 Machine Learning Module

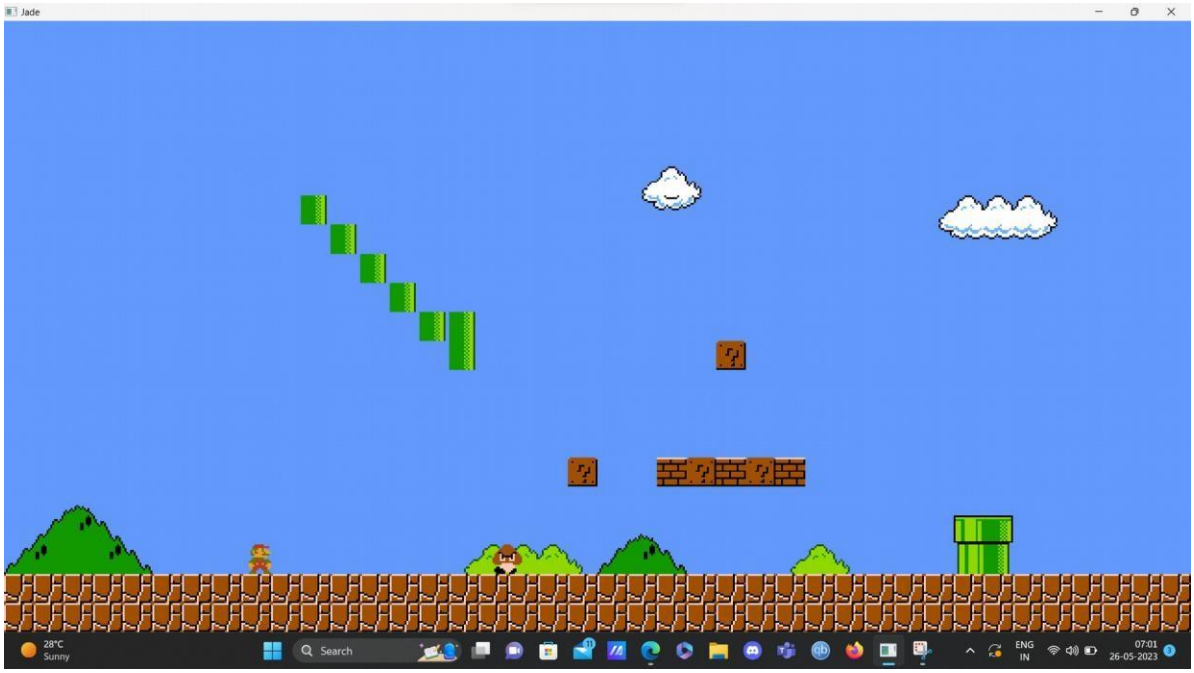
The Machine Learning Module is packaged along with the Game Engine and contains functions which can be used to easily create and use Neural Networks. The user can directly add and configure the neural network as per their needs and train it for their use. This also enables the user to map the weights and connections individually for nodes if they desire.
The ML module can also be used alongside other ML libraries and can even use Models built using other languages ex- Python by taking advantage of the functionalities provided by the library DeepLearning4j, which can load, save and utilize Keras Models.

Results:-

The primary goals and objective of this undertaking has been achieved and can be verified. The Game Engine can now be used for the development of games that can utilize the ML functionalities and create more immersive environments, smarter and more intelligent AI and NPC Agents while also being lightweight enough to run on Java. The relevant screenshots have been shown below as well as a link for the source code of the research.
The Google Drive contains a demo version of Mario, which has been packaged along with the Game Engine to setup, install and use on any machine.

Demo Screenshots and Source Code:- Link to GDrive:
https://drive.google.com/drive/folders/1Z8I2alkTZHJB0A5llwllp2OVSqxf XJ8E?usp=sharing


Jade
Blocks
Prefabs
Level File
Default
Browse
Save Level
New Level
Test Level
Layer
0
Search
ENG
IN
07:00
26-05-2023


Jade
Search
ENG
IN
07:01
26-05-2023

References

1. C. Vohera, H. Chheda, D. Chouhan, A. Desai and V. Jain, "Game Engine Architecture and Comparative Study of Different Game Engines," 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2021, pp. 1-6, doi: 10.1109/ICCCNT51525.2021.9579618. https://ieeexplore.ieee.org/document/9579618
2. 3D Game Development using LWJGL 3 - Antonio Hernández Bejarano. https://lwjglgamedev.gitbooks.io/3d-game-development-with-lwjgl/content/
3. TY - JOUR. AU - Brockman, Greg. AU - Cheung, Vicki. AU - Pettersson, Ludwig. AU - Schneider, Jonas. AU - Schulman, John. AU - Tang, Jie. AU - Zaremba, Wojciech. PY - 2016/06/05 SP - T1 - OpenAI Gym ER - . https://www.researchgate.net/publication/303822031_OpenAI_Gym
4. Applied Machine Learning for Games: A Graduate School Course Yilei Zeng, Aayush Shah, Jameson Thai, Michael Zyda University of Southern California {yilei.zeng, aayushsh, jamesont, zyda}@usc.edu. https://ojs.aaai.org/index.php/AAAI/article/view/17849/17654.
5. J. Dornig and C. Li, "Exploring the Use of Machine Learning as Game Mechanic – Demonstrative Learning Multiplayer Game Prototype," 2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), Shenzhen, China, 2020, pp. 396-399, doi: 10.1109/MIPR49039.2020.00087. https://ieeexplore.ieee.org/document/9175245
6. Alan Gavin: Blogs and Computer Concepts. https://alain.xyz/blog
7. Understanding and Implementing Neural Networks in Java from Scratch. https://towardsdatascience.com/understanding-and-implementing-neural-networks- in-java-from-scratch-61421bb6352c
8. Programming 2D Games - by Charles Kelly. http://www.programming2dgames.com/