AI CERTs®

AI+ Game Design Agent™ (1 Day)

Program Detailed Curriculum

Executive Summary

The AI+ Game Design Agent certification equips learners with essential skills for integrating artificial intelligence into game design. It focuses on leveraging AI to enhance game mechanics, player experience, and procedural content generation. This certification covers AI concepts, including machine learning algorithms, pathfinding, and behavior modeling, and demonstrates how these technologies can be applied in the development of engaging and dynamic games. Ideal for game designers, developers, and AI enthusiasts, this certification provides practical knowledge to create smarter, more adaptive gaming environments that improve both player interaction and game longevity.

Prerequisites for the AI+ Game Design AgentCourse:

- **Basic Programming Knowledge:** Familiarity with coding concepts and languages.
- **Game Design Fundamentals:** Understanding of core game mechanics and structure.
- **Mathematics and Algorithms:** Strong grasp of logic and problem-solving techniques.
- **Artificial Intelligence Basics:** Introductory knowledge of AI principles and models.
- **Creative Thinking:** Ability to envision dynamic and interactive game elements.

Module 1

Understanding AI Agents

1.1 What are AI Agents?

AI agents are autonomous entities that perceive their environment, process information, and act to achieve goals. They operate independently and adapt based on environmental changes and learned experiences.

1.2 Agent Architectures and Environments

This section explores AI agent architectures, including reactive, deliberative, and hybrid systems. It also delves into the role of environments, where agents interact and make decisions, influencing their behavior.

1.3 Decision Making and Behavior Basics

AI decision-making involves evaluating alternatives to select the best action. This section covers decision models like rule-based, utility-based, and goal-driven agents, as well as their application in game design.

1.4 Introduction to Multi-Agent Systems

Multi-agent systems (MAS) are composed of interacting agents aiming to achieve goals through cooperation or competition. This section highlights the types of interactions, including coordination, communication, and competition between agents.

1.5 Case Study: Pac-Man Ghost AI

The case study examines how the simple yet effective reactive AI of Pac-Man's ghosts creates engaging gameplay. It focuses on how each ghost's behavior design adds strategic depth to the game.

1.6 Hands On: Build a Basic Reactive AI Agent Navigating a Simple Environment Using Pygame

Participants will create a basic reactive AI agent navigating a 2D environment using Pygame. The exercise emphasizes implementing simple rule-based logic for real-time decision-making without memory.

Module 2

Introduction to AI Game Agent

2.1 What is an AI Game Agent?

AI Game Agents autonomously perform actions in games, simulating human-like behavior. They perceive environments, make decisions, and adapt, enhancing gameplay realism and player engagement.

2.2 Key Components of AI Game Agent

AI game agents consist of perception, decision-making, action execution, and learning modules, allowing NPCs to adapt, interact, and engage with the game world in meaningful ways.

2.3 Agent Architectures

Agent architectures, including reactive, deliberative, and hybrid models, define how AI agents process sensory data, plan actions, and adapt to changing environments. Each architecture impacts decision-making speed, complexity, and adaptability in games.

2.4 AI Game Agent Behaviors

AI behaviors such as patrolling, chasing, hiding, and attacking enable NPCs to respond to dynamic game environments. These actions, driven by environmental cues and decision-making logic, enhance gameplay immersion and challenge

---

2.5 Case Study: Racing Games (e.g., Mario Kart, Forza Horizon)

Explores how racing games like Forza Horizon and Mario Kart use AI agents to enhance competition, utilizing real-time environment adaptation and reinforcement learning for smarter NPC behaviors.

2.6 Hands-On: Creating a Simple Box Movement Game in Playcanvas

In this hands-on exercise, users build a simple 2D game in Playcanvas, gaining experience in implementing basic movement mechanics, handling user inputs, and understanding game interactivity for engaging gameplay

---

Module 3

Reinforcement Learning in Game Design

3.1 Basics of Reinforcement Learning

Explains key RL concepts, such as agents, environments, states, actions, rewards, and policies. Highlights how agents learn through trial and error to optimize decision-making and maximize cumulative rewards.

3.2 Key Algorithms: Q-Learning and SARSA

Introduces Q-Learning and SARSA algorithms, explaining how they enable agents to learn optimal policies through trial and error. Q-learning is off-policy, learning the best actions regardless of behavior, while SARSA is on-policy

3.3 Applying RL to Game Agents

Describes how RL can be applied to train game agents, enabling them to autonomously improve decision-making and adapt to dynamic environments, creating smarter, more interactive game AI.

3.4 Challenges and Solutions in Game-based RL

Explores challenges like balancing exploration and exploitation, managing sparse rewards, and handling large state spaces in RL for games. Offers solutions like reward shaping and deep Q-networks.

3.5 Case Study: AlphaZero in Games: Mastering Chess, Shogi, and Go through Self-Play and Reinforcement Learning

Analyzes AlphaZero's innovative use of reinforcement learning to master Chess, Go, and Shogi through self-play. It combines deep neural networks and Monte Carlo Tree Search, achieving superhuman performance without human input.

3.6 Hands On: Train a simple RL agent in OpenAI Gym environment

Provides hands-on experience with reinforcement learning by guiding users through training an agent in OpenAI Gym. The agent learns to balance a pole on a moving cart using Q-learning in the CartPole-v1 environment.

Module 4

AI for NPCs and Pathfinding

4.1 Understanding NPCs as AI Agents

This section introduces NPCs as AI-driven entities within games, exploring their decision-making, role in enhancing storylines, and interaction with the environment, contributing to dynamic gameplay and immersive player experiences.

4.2 Simple AI Techniques for NPCs

Explains basic AI methods like Finite State Machines (FSM) and Behavior Trees (BT) to simulate NPC behavior, focusing on decision-making processes for actions such as patrolling, attacking, and fleeing in games.

4.3 Pathfinding Algorithms

Covers pathfinding algorithms like A*, Dijkstra's, and BFS. These algorithms enable NPCs to find optimal routes in dynamic environments, avoiding obstacles and ensuring smooth, efficient navigation through the game world.

4.4 Obstacle Avoidance and Movement Optimization

Explores techniques for obstacle detection and avoidance, ensuring NPCs navigate around physical and dynamic barriers, while optimizing movement for smooth and efficient interactions, improving realism and player immersion in games.

4.5 Case Study

This case study examines how Halo uses Finite State Machines (FSM) and A* pathfinding algorithms to create dynamic, strategic, and immersive enemy behaviors, enhancing combat scenarios and player interaction in real-time.

4.6 Hands-On

In this hands-on exercise, you will create a basic NPC AI agent using A* pathfinding for movement and a Finite State Machine (FSM) for decision-making, enabling the NPC to chase or patrol effectively.

Module 5

AI for Strategic Decision-Making

5.1 Decision Trees and Minimax for Game AI

Decision trees and the Minimax algorithm provide fundamental methods for strategic AI decision-making. Decision trees break down decisions sequentially, while Minimax evaluates all possible moves in two-player games to choose optimal strategies, ensuring competitive gameplay.

5.2 Monte Carlo Tree Search (MCTS) for AI Agent

Monte Carlo Tree Search (MCTS) is used for AI decision-making in complex games. It combines exploration and exploitation through random sampling, simulating numerous future game states to evaluate and determine the best possible move for agents.

5.3 Utility-Based Decision Making for Game AI

Utility-based decision making empowers AI agents to evaluate various actions based on calculated utility scores. This approach allows for more dynamic and context-sensitive behavior, with actions selected based on their effectiveness in a given situation, enhancing gameplay realism.

5.4 AI in Real-Time Strategy (RTS) Games

In RTS games, AI must handle real-time decisions, balancing resource management, unit control, and strategic planning. It adapts to evolving player tactics, ensuring that NPC factions create challenging and dynamic gameplay by responding to changing battlefield conditions.

5.5 Case Study: StarCraft II AI by DeepMind

This case study explores DeepMind’s AlphaStar, which uses Monte Carlo Tree Search and reinforcement learning to master strategic decision-making in StarCraft II, competing successfully against professional human players.

5.6 Hands-On: Implement a Basic MCTS Agent for Tic-Tac-Toe Using Pygame

This hands-on exercise guides you through building an MCTS-powered Tic-Tac-Toe agent using Pygame. You'll implement Monte Carlo Tree Search to evaluate optimal moves and enhance the AI's decision-making capabilities.

Module 6

AI Game Agent in 3D Virtual Environments

6.1 3D Environment Representation and Challenges for AI Agents

AI agents must process complex 3D environments, overcoming challenges like scale, dynamic changes, sensor noise, and occlusion to navigate effectively, requiring efficient spatial models and real-time adaptability.

6.2 Navigation Mesh Generation for AI Agents in 3D

NavMeshes simplify pathfinding for AI agents in 3D spaces, enabling navigation through complex terrains. Different techniques, from manual to dynamic mesh generation, help AI agents avoid obstacles and find optimal paths.

6.3 Complex Agent Behaviors in 3D Worlds

Advanced agent behaviors in 3D environments involve intelligent decision-making, interaction with the surroundings, and collaboration with other agents, providing realistic, adaptive responses that enhance player experience and gameplay immersion.

6.4 Case Study: The Last of Us

The Last of Us uses AI agents with integrated navigation meshes and animation systems to navigate complex 3D environments, adapting to player actions, coordinating tactics, and enhancing realistic, immersive gameplay.

6.5 Hands-On: Develop a 3D AI Agent with Navigation and Interaction in Unity Using NavMesh and C#

In this hands-on exercise, design a 3D AI agent with autonomous navigation and interaction capabilities, demonstrating AI behaviors like patrolling, obstacle avoidance, and target tracking in a controlled, interactive game environment.

Module 7

Future Trends in AI Game Design

7.1 Current and Future AI Trends

Explore the evolving role of AI in game design, highlighting current advancements in machine learning, procedural generation, and adaptive gameplay, and envisioning future innovations in AI-driven gaming experiences.

7.2 The Future of Generalist AI in Gaming

Examine how general-purpose AI models will shape the future of gaming by enabling dynamic and multi-faceted gameplay, from adaptive narratives to evolving NPC behaviors, enhancing player immersion.

7.3 Case Study: No Man's Sky Procedural Generation

Investigate the use of procedural content generation in No Man's Sky, showcasing how algorithmic design powers infinite worlds and enriches player engagement, fostering exploration and replayability.

Module 8

Capstone Project

8.1 Task Description

Design and build a basic AI game agent, applying AI techniques to create an interactive agent within a 2D or 3D game environment using frameworks like Unity3D, Playcanvas, or Pygame.

8.2 Practical Implementation

Set up game engines, initialize environments, and develop the AI agent's behaviors. Utilize tools like pathfinding, obstacle avoidance, and reinforcement learning to program the agent's interactions and decision-making.

8.3 Testing and Debugging

Test the AI agent's behavior in real-time, analyze performance, validate behaviors, and optimize decision-making processes. Iterate through adjustments to improve agent performance and game dynamics based on testing outcomes.

8.4 Hands-On

Apply learned concepts to create and test a functional AI agent within a game environment. Engage in practical development, testing, and debugging, refining agent behaviors and decision-making through iterative improvements.