Departme
Engineerin

Національний технічний університет Укра
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТ
імені ІГОРЯ СІКОРСЬКОГО»

Use of Unity game application technology for radio engineering systems

Curriculum of the academic discipline (Syllabus)

Course details

Level of higher education	**First (bachelor's)**
Field of knowledge	17 — Electronics, automation, and electronic communications
Special	172 — Electronic communications and radio engineering
Educational program	Intelligent technologies of radio electronics Information and communication radio engineering Radio engineering computerized systems
Status of the discipline	Elective
Form of study	Full-time (day)
Year of training, Semester	3rd year, 5th semester
Scope of the discipline	120 hours (16 hours – Lectures, 30 hours – Laboratory work, 74 hours – Independent work)
Semester control/control measures	Credit
Class schedule	
Language of instruction	Ukrainian
Information about the course leader/teachers	Lecturer: Ph.D., Associate Professor Pavlo Yuriyovych Katin, ka72tin@gmail.com mobile +38(098)202-08-11 Laboratory: Ph.D., Associate Professor Pavlo Yuriyovych Katin
Location of the document	

Curriculum

1. Description of the academic discipline, its purpose, subject matter, and learning outcomes

Description of the discipline. The main objective of the discipline "Development of Game Applications" is to provide systematic knowledge about:

- creating prototypes of radio engineering systems in the form of emulation based on a game engine;
- the basics of programming languages for script development;
- the basics of the game engine interface and the integrated game development environment;
- basic approaches and skills for creating games and game scenes in two-dimensional and spatial versions;
- types, technologies, software platforms, and integrated development environments for game applications using Unity as an example;
- organization and composition of a development team, monetization methods, related game application technologies;
- software architecture and modern technologies for game applications (GA);
- the minimum required knowledge of programming languages for developing GA;
- the basics of the software architecture of the GA "engine";
- the structure and general principles of GI design;
- elements of GI logic and typical programming patterns.

In addition, the discipline provides skills in developing GA using the selected GA technology and game engine. The syllabus of the discipline includes educational tasks and programming practices that students need to achieve learning outcomes, taking into account the special regime.

There is a standard approach to teaching programming, from simple to complex, which simplifies the process of further learning. The course is designed for students who do not know a programming language for developing game scripts.

The final task is to develop a prototype of an intelligent system. To complete this task, students use theoretical knowledge and apply practical skills acquired throughout the lecture course and laboratory work.

Subject of the discipline: the process of developing IS and prototypes of radio engineering systems based on a game engine.

Interdisciplinary connections: Prerequisite disciplines**:** Fundamentals of Programming; Discrete Mathematics, Physics.

Supporting disciplines: Bachelor's project, Master's thesis (in matters of selection, development, and justification of decisions).

Purpose of the course: To train developers and managers in the field of IS with the aim of creating prototypes of radio engineering systems based on a game engine.

Learning outcomes are Knowledge:

- options for creating radio engineering system emulations based on game engines;
- types of game applications, their features, history, ratings, commercial level, and monetization methods;
- the basics of mathematical support and elements of artificial intelligence in IS development;

- modern technologies of game applications, advantages and disadvantages of different technologies;
- the basics of programming languages used as scripts for developing game applications, the advantages and disadvantages of different programming languages;
- proficiency in one of the programming languages for game development, typical patterns used in the game industry;
- basics of software architecture of the game engine, different types of engines, their advantages and disadvantages, thorough knowledge of one of them;
- mechanisms for building a graphical interface and design;
- basics of computer game logic;
- basics of release preparation technology.

Skills:

- develop a prototype of a radio engineering system emulation in a team led by an experienced manager and developer (junior level);
- independently develop a computer game prototype using one of the selected technologies;
- develop a design, manage the development of a computer game design;
- monetize solutions, test and support the developed IS.

2. Prerequisites and post-requisites of the discipline (place in the structural-logical scheme of training under the relevant educational program)

Prerequisites: Fundamentals of programming; knowledge of one of the programming languages that form the basis of game engine scripts (C#, C++, C); basic knowledge of discrete mathematics and analytical geometry, knowledge of English in accordance with the program, knowledge of physics.

Post-requisites: Theoretical knowledge and practical skills acquired during the course (credit module) necessary for the development of a bachelor's project.

3. Course content

Lectures.

<u>Topic 1.</u> Fundamentals of computer engines and types of computer games.

Lecture 1. Types of computer games and technologies for their development using Unity (Unreal 5) as an example. Types of computer games. Personal safety issues when interacting with game programs. Game engines and game application development technologies. Preparing a game project repository.

<u>Topic 2.</u> Creating a game concept and conducting a business analysis

Lecture 2. Basics of planning, organization, and analytics for game application development and radio engineering system emulation systems.

Computer game genres. 3D and 2D directions. Creating emulation systems using a game engine. Dividing into project teams and determining the direction of the game. Basics of analytics. Game concept development. Decomposition into mechanics. Description and formalization of Core Gameplay and Meta Gameplay. Project creation process. Connecting tools and getting acquainted with the

Asset Store. Project file organization and scripting basics.
Topic 3. Basics of the integrated development environment for game applications and script programming.

Lecture 3. Basic elements of an integrated game application development environment and use of data types.
Basics of the integrated script development environment. Basic script pattern of the Unity game engine
(Unreal 5). Connection between elements of the integrated game application environment and game script using the example of terminal output. Differences between version 2022.3.62f LTS (and earlier versions) and version 6000.2.2f1 LTS. Basics of variables (fields), data types, basic concepts of methods. Basics of control flow.
Topic 4. Software architecture of game engine scripts.

Lecture 4. Basics of basic script architecture using game engine methods as an example.
Game character control. Basics of game engine design and scripting architecture using the example of Unity (Unreal 5) game character control in a simple 3D (2D) scene.
Debugging script source code using console output. Declaring variables of different types and initializing them. Using access modifiers, features of access modifiers in the context of game engine usage. Understanding the scope of variable visibility. Basics of declaring, defining, and calling a method within a single class. List and reference information about typical methods of the basic game engine pattern.
Topic 5. Software component of a computer game based on an engine (Unity, Unreal Engine 5) based on a
simple 3D (2D) scene.

Lecture 5. Basics of flow control in a game script and loop creation operators.
Flow control operators in a program (game engine script). Loop organization operators. Additional elements for loop operators. Entity containers and collections.
Topic 6. Basics of object-oriented programming in game engine scripting systems.

Lecture 6. Game engine classes.
Declaration, description, creation of a class instance and structure. Using methods, properties, and other class members. Using structure fields. General information, reference syntax. Basic principles of object- oriented programming. Features of OOP in game engines.
Topic 7. Using a game engine to create components of a game or radio engineering system emulator.

Lecture 7. Creating a game (emulation) level and scene in a game engine.
Fundamentals of game design and emulation. Technologies for creating a game level. Game objects and prefabs. Software control of lighting emulation. Animation of game characters and the environment.
Topic 8. Software control of game and non-game characters in a computer game scene.

Lecture 8. Scene with game character control.
Software control of a game character model in the context of movement and rotation. Processing player activity through keyboard, mouse, and other input devices at the script level. Programming the game view from the player's perspective (programmable video camera behavior). Physics simulation in the game engine. Basic game colliders and triggers.
Topic 9. Game mechanics and artificial intelligence of non-player characters.

Lecture 9. Perfect components of a computer game and artificial intelligence.
Programming jumps. Using layers in the game system. Creating instances of prefabs and objects. The software

component of the game manager. Computer user interface.

Games. Game engine navigation system and navigation networks. Artificial intelligence of non-player characters.

Laboratory classes.

Name of laboratory work
Laboratory work 1. * Researching the basic pattern of the Unity game engine using the example of two-dimensional technology
Laboratory work 2. * Researching the basic pattern of the Unity game engine using the example of a three-dimensional game application
Laboratory work 3. Researching typical patterns in game application scripts
Laboratory work 4. Documenting the game concept
Laboratory work 5. Basic components of a game application
Lab work 6. Business component of a game application
Laboratory work 7. Network elements of a game application
Lab 8. Fundamentals of artificial intelligence in games and game user interfaces

4. Teaching materials and resources

Basic literature

1. Learning C# Programming with Unity 3D by Alex Okita.
2. Unity in Action: Multiplatform Game Development in C# with Unity 5 by Joe Hocking.
3. Introduction to Game Programming: Using C# and Unity 3D by Vahe Karamian.
4. Learning C# by Developing Games with Unity 3D Beginner's GuideSep by Terry Norton.
5. C# Game Programming Cookbook for Unity 3D by Jeff W. Murray.

Supplementary literature

1. Unity 3D UI Essentials by Simon Jackson.
2. Game Programming Patterns by Robert Nystrom.
3. Creating E-Learning Games with Unity by David Horachek.
4. Learn Unity for 2D Game Development by Alan Thorn.
5. Unity for Architectural Visualization by Stefan Boeykens.
6. Removed
7. Removed
8. https://unity.com
9. https://www.unrealengine.com/en-US/
10. https://blog.studica.com/how-to-setup-github-with-unity-step-by-step-instructions
11. https://uk.wikipedia.org/wiki/Жанри_відеоігор

12. https://dtf.ru/gamedev/5375-opisanie-ciklov-igry-primery-i-sovety

13. https://www.epravda.com.ua/rus/publications/2018/09/11/640413

14. https://www.epravda.com.ua/publications/2021/11/26/680153/

15. https://docs.unity3d.com/ru/530/Manual/ExecutionOrder.html

Educational content

6. Methodology for mastering the academic discipline (educational component)

Lectures (18 hours)

No	Lecture topic and list of main questions (list of teaching aids, references to literature, and assignments for independent study)
1	Topic 1. Fundamentals of the computer engine and game components
	Studying the graphical interface of the game engine. Preparing the repository, flow, and task manager. Creating a project. Connecting tools and getting acquainted with the Asset Store. Organizing project files. *Implementation of 1 scene using the example of a 3D game based on a ready-made screenshot provided by the RigitBody component. Unity documentation example RigitBody. No scripting.
2	Topic 2. Preparation and formalization of the basics of the game concept and the basics of business analysis.
	Computer game genres. 3D and 2D directions. Division into project teams and determination of the game direction. 3D and 2D directions. 2. Basics of analytics. Game concept development. Decomposition into mechanics. 3. Description and formalization of Core Gameplay and Meta Gameplay. 4. Basic model issues and monetization solutions. 5. Basics of programming.
3	Topic 3. Mathematical models of the Unity engine (or Unreal Engine, if preferred) and implementation physical models.
	Life cycle of mono behavior. Direct script wake-up connections, such as creating connections. Basics of creating and configuring basic elements of the control system Creating and configuring the game character scene and configuring software entities and other objects in the 3D system. Basics of script control of the game character. Basics of control, scene loading, control of graphic and sound entities from the player's side using software scripts based on the C# language. Primitive graph and images.

	Selecting and creating basic architectural pattern scripts. MVC, ECS.
4	Topic 4. Mathematical models of the Unity engine (or Unreal Engine, if you prefer) and implementation physical models.
	Game space. A simple scene with a platform and a game character (using a primitive or a 2D image as an example). Spatial mathematical models of the Unity engine based on linear algebra and analytical geometry. Vector mathematics. Physical models in Unity Physics. Software configuration and control of physical model parameters. Implementation of motion models, rotations, ballistic calculations. Practical implementation of software architecture based the basis of standard patterns (templates).
5	5. Implementation of game entities based on mathematical and physical modeling in the Unity game engine (or Unreal Engine, if preferred).
	A single scene with a game character (using a 3D primitive or a 2D image as an example). Game logic. Game mechanics. Event. Pattern strategy. Creation of Unity game entity systems and modules based on linear algebra, analytical geometry, and physical models. Examples of game mechanics implementation. Implementation of game architecture. Practical examples based on game solutions.
6	Topic 6. Perfecting the appearance of a game character in a 3D (2D) animation system 3D (2D) animation system and game meta-mechanics.
	State pattern, state machine. Graphical and software implementation and typical settings for game character software entities in a 3D system. Script control of game characters. Graphical and software implementation and typical settings for NPC (Non-Player Character) and other objects. Features of script control. Features of animation, Animator Controller. Scripted animations using the DOTween engine as an example. Basics of creating game meta-mechanics. Leveling up, levels, characteristics. Saving the game. *Perfect version control system.
7	Topic 7. Perfecting game space settings in a 3D (2D) system and testing

	.
	Working with Terrain and configuring the environment. Placing the game environment. Configuring the game character's camera in a 3D (2D) system. The need for and features of using multiple video cameras. Lighting settings, lighting implementation features, light "baking." 2D camera settings. Implementing testing in the project. Creating a test plan. Debugging testing processes. Planning module testing. Test design techniques. Creating test cases and checklists. Direct module testing. Pre-release and post- release testing. Vectors development of a tester: automation testing, testing performance testing, ethical hacking.
8	Topic 8. Artistic design of the game space in a 3D system and creation of the musical emotional component and monetization of the game
	Working with Particle System. Configuring shaders and 3D (2D) materials. Post-processing. Technology for adding sound effects and audio mixers to the game. Implementing in-game purchases and subscriptions. Implementing advertising. Integrating analytics. Game metrics and analysis.
9	Topic 9. Fundamentals of artificial intelligence in games and game user interfaces.
	Basics of working with the interface. Creating main windows. Configuring interface elements. Interface and mechanics connection. Implementation of the basics of game character artificial intelligence in a 3D system. State machine. Implementation of the basics of artificial intelligence for NPC (Non-Player Character) software entities and other objects. Features of script control. Saving game releases. Basics of working with App Store, Play Market, and Steam services. Organization of work with the review market.

Laboratory classes (With the consent of the instructor, it is permitted to defend projects received (developed) during the courses when completing LR1-2)

No	Name of laboratory work	Number of aud. hours
1	Laboratory work 1. * Researching the basic pattern of the Unity game engine Unity using the example of two-dimensional technology	4
2	Laboratory work 2. * Researching the basic pattern of the Unity game engine Unity using the example of a three-dimensional game application	4
3	Laboratory work 3. Researching typical patterns in game application scripts of a game application	4
4	Laboratory work 4. Documenting the game concept	4
5	Laboratory work 5. Basic components of a game application	4
6	Laboratory work 6. Business component of a game application	4
7	Laboratory work 7. Network elements of a gaming application	6
8	Laboratory Work 8. Fundamentals of Artificial Intelligence in Games and User Interfaces user interface	6

7. **Independent work of a student/graduate student (66 hours)**

The discipline is based on independent preparation for classroom sessions on theoretical and practical topics.

No.	Area of independent preparation	Number of hours	Literature
1	Preparation for lecture 1	1	1
2	Preparation for laboratory work 1	1.5	1
3	Preparation for lecture 1	1	1
4	Preparation for lecture 2	1	1
5	Preparation for laboratory work 2	1.5	1
6	Preparation for lecture 2	1	1
7	Preparation for lecture 3	1	1
8	Preparation for Laboratory Work 3 (part 1)	1.5	1
9	Preparation for lecture 3	1	1
10	Preparation for lecture 4	1	1
11	Preparation for lab work 3 (part 2)	1.5	1
1	Preparation for lecture 4	1	1
13	Preparation for lecture 5	1	1
14	Preparation for laboratory work 4 (part 1)	1.5	1

1	Preparation for lecture 5		1	1
16	Preparation for lecture 6		1	1
17	Preparation for lab work	4 (part 2)	1.5	
18	Preparation for lecture 7		1	
19	Preparation for Lecture 7		1	
20	Preparation for laboratory work	5	1.5	
21	Preparation for lecture 8		1	1
2	Preparation for Lecture 8		1	1
23	Preparation for laboratory work	6 (part 1)	1.5	1
24	Preparation for lecture 9		1	1
25	Preparation for Lecture 9		1	1
26	Preparation for laboratory work	7, 8 (part 2)	1	1
27	Preparation for the midterm test		4	1
28	Preparation for the exam		30	1-5
29	Getting started with MVS		2	1
30	Install MVS		1	1
31	Essential C# Features for Web Development. Stage 1.1		1	1
32	Essential C# Features for web developing. Stage 1.2		1	1
33	Essential C# Features for web developing. Stage 2.1		1.5	1
34	Essential C# Features for web developing. Stage 2.2		1	1
35	Essential C# Features for web developing. Stage 3.1		1	1
36	Essential C# Features for web developing. Stage 3.2		2	1
37	Essential C# Features for web developing. Stage 4.1		2	1
38	Essential C# Features for web developing. Stage 4.2		1	1
39	Essential C# Features for web developing. Stage 5		1	1
40	Essential C# Features for web developing. Stage 6		2	1

Policy and control

8. Academic discipline (educational component) policy

Attendance at lectures is mandatory. In exceptional circumstances, attendance requirements and other aspects of the policy may be subject to change.
Attendance at computer lab classes may be sporadic and, if necessary, for consultation/defense of laboratory work.
Rules of conduct in class: be active, respect others, turn off phones. Follow the academic integrity policy.

Rules for defending laboratory work: work must be done in accordance with the tasks set and according to the option.

9. **Types of assessment and the learning outcomes assessment rating system (LOAS)**

During the semester, students complete 8 laboratory assignments. The maximum number of points for each assignment is 10.

Points are awarded for:

1. quality of laboratory work: 0-5 points;
2. answers during the defense of laboratory work: 0-3 points;
3. timely submission of work for defense: 0-2 points. Criteria for assessing the quality of performance:

5 points – work performed qualitatively, in full;
4 points – the work is done well, in full, but has some flaws;
3 points – the work is completed in full, but contains minor errors; 2 points – the work is completed in full, but contains significant errors; 0 points – the work is not completed in full.

Answer evaluation criteria:
3 points – the answer is complete and well-reasoned;
2 points – the answer is correct, but has shortcomings or minor errors; 1 point – the answer contains significant errors;
0 points – no answer or the answer is incorrect.

Criteria for assessing the timeliness of the submission of the work for defense: 2 points – the work is submitted for defense no later than the specified deadline;
0 points – the work is submitted for defense after the specified deadline.

Maximum number of points for completing and defending laboratory work: 10 points × 8 lab sessions = 80 points.

During the semester, **quizzes on the topic of the current lesson** are held during lectures. Maximum number of points for all quizzes: 3 points. The number **of quizzes on the topic of the current lesson** for one student is unlimited.

The module test consists of 1 theoretical and 1 practical question. The answer is graded on a scale of 0 to 20 points. The criteria for evaluating each question of the test are as follows: 9-10 points – the answer is correct, complete, and well-reasoned;
7-8 points – the answer is correct, detailed, but not very well argued; 5-6 points – the answer is generally correct, but has shortcomings;
3-4 points – the answer has minor errors;
1-2 points – the answer contains significant errors; 0 points – no answer or incorrect answer.

Maximum number of points for the module test: 10 points × 2 questions = 20 points.

The rating scale for the discipline is as follows:
R = $_{RC}$ = 80 points + 20 points = 100 points.

Calendar control: conducted twice per semester as monitoring of the current status of syllabus requirements fulfillment.

At the first assessment (week 8), the student receives a "pass" if their current rating is at least 15 points (50% of the maximum number of points a student can receive before the first assessment).

At the second assessment (14th week), the student receives a "pass" if their current rating is not less than 20 points (50% of the maximum number of points that a student can receive before the second assessment).

Semester control: credit

Conditions for admission to semester control:

With a semester rating ($_{RC}$) of at least 60 points and all computer workshop assignments completed, the student automatically receives a credit according to the table (Table of correspondence of rating points to grades on the

university scale). Otherwise, they must complete a credit test.
A prerequisite for admission to the credit test is the completion and defense of laboratory work.
If a student does not agree with the automatic grade, they can try to improve their grade by writing a credit test, in which case their points earned during the semester are retained, and the better of the two grades received by the student is awarded (a "soft" grading system).
Table of correspondence between rating points and grades on the university scale:

Number of points	Grade
100-95	Excellent
94	Very good
84	Good
74-65	Satisfactory
64-60	Sufficient
Less than 60	Unsatisfactory
Admission requirements not met	Not admitted

10. **Additional information on the discipline (educational component)**

Various operating systems can be used for work at the student's discretion and in agreement with the instructor.
If students wish to use other programming languages or technologies in the computer lab, they can be included as an additional element in the basic architecture of the distributed web application.

Work program for the academic discipline (syllabus):

Compiled by Associate Professor of the Department, Ph.D., Associate Professor, P. Yu. Katin.

Approved by the Department of Radio Engineering (Minutes No. 06/2025_ dated 24.06. 2025).

Approved by the Methodological Commission of the Radio Engineering Faculty (protocol No. 06/2025 dated 26.06.2025)

Appendix 1

Program for preparation for the final interview.

1. Basic elements of the Unity3d development environment.
2. Textures and materials. Landscape editor management.
3. Techniques and practices for creating textures.
4. Methods and practices for adding new textures to a project.
5. Basics of creating a landscape on Terrain, give examples.
6. Technology for placing the player on the landscape.
7. Example of script creation.
8. Applying scripts to a game character.
9. Applying scripts to scene elements.
10. Script patterns in game applications.
11. Architecture and example of using the Command pattern in game applications.
12. Behavior patterns in the context of game applications. Architecture and example of use in game applications Bytecode.
13. Architecture and example of using the Flyweight pattern in gaming applications.
14. Behavior patterns in the context of gaming applications. Architecture and example of using the Subclass Sandbox pattern in gaming applications.
15. Architecture and example of using the Prototype pattern in gaming applications.
16. Behavior patterns in the context of game applications. Architecture and example of use in gaming applications of the Type Object pattern.
17. Architecture and example of use in gaming applications of the Singleton pattern.
18. Behavior patterns in the context of game applications. Architecture and example of use in game applications of Decoupling Patterns.
19. Justification for using the State pattern in gaming applications.
20. Architecture and example of using the State pattern in gaming applications.
21. The structure of Game Loop pattern scripts.
22. The structure of Double Buffer pattern scripts.
23. Structure of Update Method pattern scripts.
24. Examples of basic movement scripts in Unity3d.
25. Creating assets and using ready-made assets in the Asset Store.
26. Methods for developing and testing scripts for game characters. Triggers.
27. Features of working with CharacterController.
28. User interface and game menu creation.
29. Animation. Using ready-made animations and creating your own animations.
30. Basics of working with 2D games. Technology for creating 2D games.
31. Basics of working with sprites in Sprite Editor.

32. Computer game genres. The psychology of computer games. Analysis of cult games.