aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

2020 Subject & Assessment Guide

Artificial Intelligence for Games

ICT50215

Diploma of Digital and Interactive Games

academy of interactive entertainment
Sydney Melbourne Canberra Adelaide Lafayette Seattle Online

Table of Contents

Artificial Intelligence for Games

Units of Competency

The units of competency that are covered in this subject are as follows:

ICTGAM404 – Apply artificial intelligence in game development

ICTICT408 – Create technical documentation

Assessment processes and competency evidence requirements are described in the *Assessment Criteria* section below. If you have prior or other evidence against competency you should discuss this with your teacher.

Subject Overview

Overall Learning Outcomes

- Develop and apply an understanding of modern A.I. techniques for games
- Implement pathfinding algorithms
- Implement decision making techniques

Subject Description

Artificial Intelligence (A.I.) in games is a broad topic. A.I. covers a range of techniques that allow the computer to think and act based on a set of rules. The aim of most A.I. is to succeed at whatever its goal is. In the case of game A.I. its goal is actually to lose, but to lose convincingly and encourage the player to overcome a challenge.

Brian Schwab, author of "**AI Game Engine Programming**", says the aim of game A.I. is "to be a good dad". Imagine an A.I. entity playing a game with their kid. A bad dad would defeat their kid as quick as possible. A good dad would encourage their kid to rise to the challenge and overcome it, adapting the difficulty to meet the skills of their child.

This subject will teach you various decision-making algorithms and techniques used in games, along with locomotion techniques to move entities around your game worlds. You will also learn pathfinding algorithms that help A.I. find their way to a goal, be that a location within the level or a certain goal state such as world domination!

Industry Relevance

Artificial intelligence is a big part of modern video games that need competitive opponents and various non-player characters. Games are full of little smart agents making all kinds of decisions, from

humanoids to little rodents that need to scamper around environments realistically. Simulation industries also require A.I. as does film. The Lord of the Rings movie franchise made heavy use of A.I. techniques for their hordes of orcs and humans battling it out on the big screen.

Assumed Knowledge

- Knowledge of C++ programming sufficient to create complex real-time applications
- Knowledge of basic vector and matrix mathematics for 3-D coordinate systems

Subject Textbooks

Although not required, the following textbooks are recommended to aid in the completion of this subject:

- Funge, J, Millington, I, ***Artificial Intelligence for Games***, 2nd Edition, CRC Press (2009)
- Mark, D, ***Behavioral Mathematics for Game AI***, Charles River Media (2009)
- Buckland, M, ***Programming Game AI by Example***, Wordware Publishing Inc (2005)
- Schwab, B, ***AI Game Engine Programming***, 2nd Edition, Cengage Learning (2008)

Assessment Criteria

Assessment Description

Assessment Milestones

Please refer to your Class Schedule for actual dates on your campus

General Description

This subject will teach you various aspects of artificial intelligence for games. To be assessed as competent you will need to demonstrate a real-time simulation, which may or may not be interactive, but which demonstrates a few key assessable elements in game A.I.

The application you develop for your assessment may be a game, or an extension to a previous assessment, but it must implement the following required assessable elements:

- Entities using decision making techniques
- Entities using suitable pathfinding algorithms

Examples of decision-making techniques that are acceptable include state machines, blackboards, decision trees, behaviour trees, and planning algorithms. Suitable pathfinding algorithms include Dijkstra's algorithm, A-Star pathfinding, and navigation meshes. Other algorithms also exist and you are free to use another or a combination of algorithms to implement your simulation or game.

This assessment has been divided into three phases: a preproduction phase, the implementation phase, and a postproduction phase. Each phase consists of separate deliverables and it is recommended you schedule your work so that each deliverable can be assessed in turn.

Pre-Production: Artificial Intelligence Behaviour Plan

The pre-production phase of this assessment requires the creation of an Artificial Intelligence Behaviour Plan, to be specified with a *Technical Design Document*. The specific requirements to achieve competency for this piece of evidence are detailed in the table *Assessment Tasks and Evidence Descriptions* below.

Before implementing your techniques and algorithms you are to first create a *Technical Design Document* that discusses and lists the implementation details of the A.I. algorithms and techniques you will be using in your program, including a diagram specifying the decision-making behaviour.

For example, if you were to use Behaviour Trees as a decision-making technique then you will need to include a behaviour tree diagram that demonstrates the behaviour and logic to be implemented.

If you are unsure of what to implement your teacher may provide you with guidance and suggested simulations to create. Suggested project briefs have also been provided for you and are included in *Appendix 1*.

A template for the *Technical Design Document* has been provided for you in *Appendix 2*.

Your document must discuss the following:

- The pathfinding algorithm you will implement
- AI strategies for NPCs for the game design that are technically feasible, respond to the brief, and provide creative solutions to all design issues

Implementation: Implementing Artificial Intelligence Behaviour

The implementation phase of this assessment requires the implementation of Artificial Intelligence behaviour in a simulation or game. The specific requirements to achieve competency for this piece of evidence are detailed in the table *Assessment Tasks and Evidence Descriptions* below.

Once your *Technical Design Document* detailing your project plan has received approval from your teacher you may begin implementation.

As noted above, you must ensure that your final simulation or game contains the following assessable elements:

- Implement an NPC AI strategy
- Implement a path-finding algorithm

It is recommended that your solution is developed in C++ using the RayLib or AIEBootstrap framework, or another C++ graphics framework. While you are encouraged to use your own maths library, use of an alternative maths library (such as glm) is acceptable.

Use of alternative programming languages or full-featured game engines is strongly discouraged as it will generally be much more difficult to prove competency in writing the required algorithms. All classes and tutorials for this subject will be taught in C++ using the RayLib or AIEBootstrap framework.

Post-Production: Artificial Intelligence Behaviour Evaluation

The post-production phase of this assessment requires an evaluation of your Artificial Intelligence implementation. The specific requirements to achieve competency for this piece of evidence are detailed in the table *Assessment Tasks and Evidence Descriptions* below.

Once you have implemented your simulation or game, you are to create a short, written document that describes:

- How easy or difficult it was for you to implement your planned techniques?
- What the computational performance of your techniques was and how they could have been improved?

You must note any adjustments and modifications needed on your initial plan for the final implementation.

Evidence Specifications

This is the specific evidence you must prepare for and present by your assessment milestone to demonstrate you have competency in the above knowledge and skills. The evidence must conform to all the specific requirements listed in the table below. You may present additional, or other evidence of competency, but this should be as a result of individual negotiation with your teacher.

Your Roles and Responsibilities as a Candidate

- Understand and feel comfortable with the assessment process.
- Know what evidence you must provide to demonstrate competency.
- Take an active part in the assessment process.
- Collect all competency evidence for presentation when required.

This table defines the individual requirements for each part of the assessment criteria. Listed here are the cumulative requirements for all assessment items. The evidence requirements for specific assessment items can be seen by referring to the table listed for that assessment item in the following sections.

Assessment and Competency Requirements

1. Artificial Intelligence Behaviour Plan
Evidence that includes:
• Creation of a written document, including diagram(s), that clearly describes the artificial intelligence behaviour(s) and decision making to be implemented within a real time application
  ○ Must describe the technique and algorithms to be used for implementing the behaviour(s) and the decision-making process
  • Discusses the third-party graphical framework to be used, including:
  • Evaluation of the suitability of the framework for use in the application
  • Any technical impact of the framework has on the project design
  • Identification of the framework license, including identifying any vendor licensing issues
  ○ This document, including diagram(s), must be created before implementing the artificial intelligence behaviour(s)

2. Implement Artificial Intelligence Behaviour
Evidence that includes:
• Creation of a real time application that demonstrates the following:
  ○ Game entities displaying the implemented artificial intelligence behaviours and decision-making techniques as specified in the Artificial Intelligence Behaviour Plan
  ○ Game entities using a suitable pathfinding search algorithm as specified by your instructor
• You must submit your application in executable form that can run external to any IDE without any errors
• Submit the source code and assets for the real time application for review

3. Artificial Intelligence Behaviour Evaluation
Evidence that includes:
• Artificial Intelligence Behaviour Plan submitted with edits and improvements based on necessary modifications and amendments found during the implementation of said Artificial Intelligence Behaviour Plan
• Creation of a written document that details:
  ○ Ease of implementing the Artificial Intelligence Behaviour
  ○ Performance of the Artificial Intelligence Behaviour
  ○ Improvements that could be made to the Artificial Intelligence Behaviour

4. Followed Good Coding Practices Evidence that includes: • Projects which contain no warnings and no compile error messages when building application in debug or release • Code which follows consistent naming conventions • Source files are commented to an acceptable industry standard as specified by your instructor
5. Application Handover Evidence that includes: • Visual Studio solutions and projects that compile without errors   ○ All temporary and built executable files in the obj and bin folder have been removed • A “readme” or client document explaining how to compile, run and operate the program, for each application made • All submitted material archived in a single compressed file (zip, rar, or 7z)

Assessment Instructions for Candidate

METHOD OF ASSESSMENT

Assessment is a cumulative process which takes place throughout a subject. A ‘competent’ or ‘not yet competent’ decision is generally made at the end of a subject. Your assessment will be conducted by an official AIE qualified assessor. This may be someone other than your teacher. The evidence you must prepare and present is described

above in this assessment criteria document. This evidence has been mapped to the units of competency listed at the beginning of this document. Assessments will be conducted on a specific milestone recorded above in this assessment guide document.

ASSESSMENT CONDITIONS

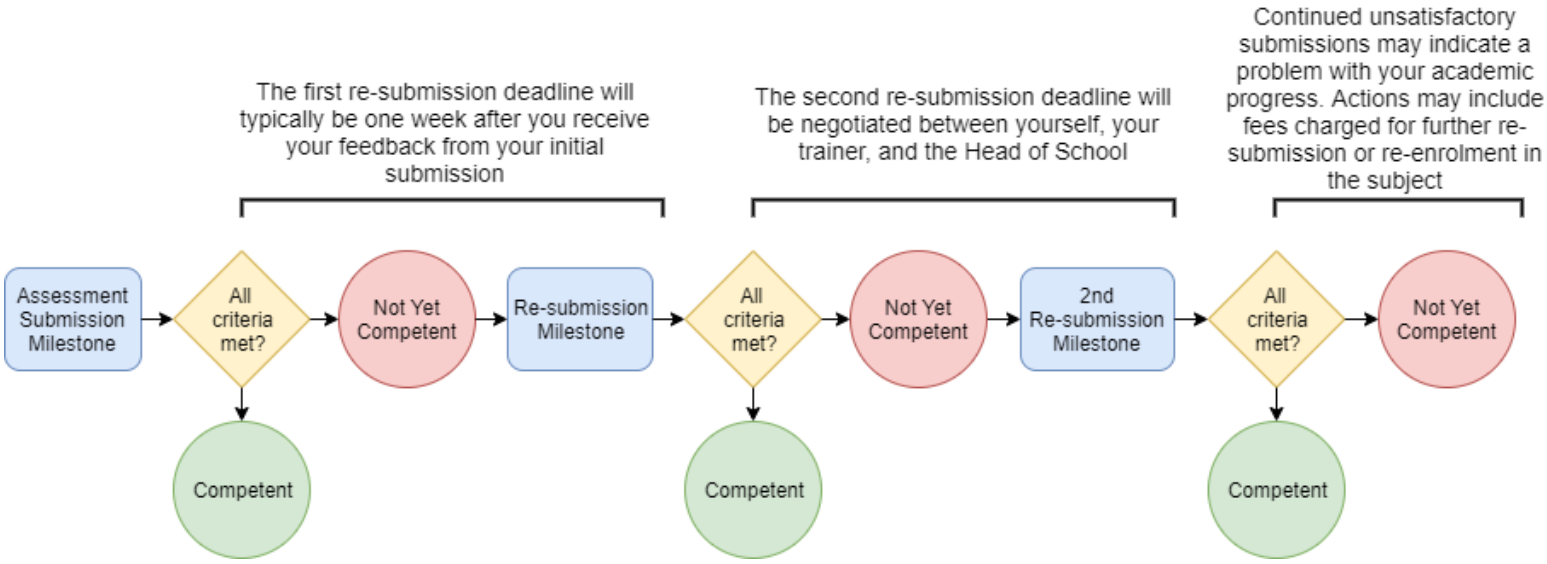
Formative assessment takes place as your teacher observes the development of your work throughout the subject and, although the assessor is likely to be aware of the evidence you are submitting, it is your responsibility to be prepared for the interview where a competency judgement is made (summative assessment). Forgetting something, or making a small mistake at the time of the milestone assessment, can be corrected. However, the assessor may choose to assess other candidates who are better prepared and return to you if time permits.

Upon completion of the assessment you will be issued with feedback and a record of the summative assessment and acknowledge that you have received the result. If you are absent for the nominated assessment milestone (without prior agreement or a sufficiently documented reason) you will be assessed as not yet competent.

GRADING

The assessment you are undertaking will be graded as either *competent* or *not yet competent*.

REASSESSMENT PROCESS



If you are assessed as being not yet competent you will receive clear, written and oral feedback on what you will need to do to achieve competence. Failing to submit an assessment will result in you being assessed as not yet competent. You will be given a reassessment milestone no more than one (1) week later to prepare your evidence. If you are unsuccessful after your reassessment, you may be asked to attend a meeting with your Head of School to discuss your progress or any support you may need and further opportunities to gain competency.

REASONABLE ADJUSTMENTS

We recognise the need to make reasonable adjustments within our assessment and learning environments to meet your individual needs. If you need to speak confidentially to someone about your individual needs, please contact your teacher.

FURTHER INFORMATION

For further information about assessment and support at AIE, please refer to the assessment and course progress sections of your student handbook.

Software

Core

Microsoft Visual Studio

Microsoft's Visual Studio is the recommended IDE for this subject. Other IDEs may be employed if desired as the content of this subject is designed to be cross-platform and IDE agnostic, however we cannot guarantee that all subject material will operate as intended on other IDEs and platforms.

- https://www.visualstudio.com/

References and additional material

Artificial Intelligence for Games

- https://en.wikipedia.org/wiki/Artificial_intelligence_in_video_games
- http://aigamedev.com/
- http://www.gameai.com/
- An Intro to Video Game AI for Beginners and Youn Programmers (youtube)

Appendix 1

The following game briefs have been designed around the specific, cumulative evidence you must prepare for and present by your assessment milestone to demonstrate you have competency in the requisite knowledge and skills for this subject. Submissions conforming to one of these briefs will provide the specific evidence listed in the table titled *Assessment and Competency Requirements* at the end of this assessment item.

For this assessment you may present an alternative game, provided the requisite performance criteria are met. However, any deviation from the provided project briefs should be the result of individual negotiation with your teacher to ensure your final assessment will meet the evidence requirements of this assessment.

Game Project Brief 1: Forest Simulation

Overview:

Create a 2D top-down non-interactive simulation of a forest.

Your forest should include carnivores and herbivores (i.e., predators and prey), and each animal will have a base set of needs – for example, hunger, thirst, and tiredness.

Implement decision making algorithms (like state machines, decision trees or behaviour trees) so that your animals can make decisions based on their current needs.

When an animal determines that is thirsty, for example, it can seek a path that will take it to its nearest water source.

Predatory animals should wander the map looking for prey and chase them when encountered.

Implementation:

Create a simulation with at least two different animals, at least one type of predator and one type of prey.

Both predatory and prey animals should have base parameters that include (but may not be limited to) the following:

- Health
- Thirst
- Hunger
- Tiredness

These parameters should update in a realistic way as the simulation progresses. For example, an animal's hunger will gradually increase over time and reset to zero once the animal eats.

Implement a decision-making algorithm so that an animal will choose the best (most appropriate) action to perform given its current parameters. For example, if an animal is hungry and all other needs are low, then it would choose to eat. An appropriate algorithm to use would be a decision tree or behaviour tree, however a state machine may also be possible depending on your implementation.

When an animal identifies an action to perform (for example, to drink) and the completion of that action requires traversing the map (for example, going to the water hole), the animal should use a path finding algorithm to find an appropriate (not necessarily the shortest) path across the map.

When predatory animals are hunting, they should use the wander behaviour (or another appropriate algorithm) to search the map looking for prey. When prey is detected both animals may switch to behaviours (like seek and flee) to navigate rather than choosing a specific point on the map and pathfinding.

Optional Additions:

Feel free to expand on these base requirements and improve your simulation in any way you prefer.

Additional features you may wish to add could include:

- Blackboards:
  When a prey animal detects a predator it writes a message to the blackboard. Any other animal nearby could read the message on the blackboard and be aware of the predator even if they are not able to detect it directly.
- Flocking:
  Have your prey animals move in herds to minimize the rise of any individual being attacked by a predator.
- Other game entities:
  Implement other entity types, like the scavenger which will feed off the prey killed by the predators. Scavengers would 'hunt' by following the predators, remaining close enough to identify uneaten kills but still remaining far enough away that they don't become the predator's prey.

Game Project Brief 2: Stealth Game

Overview:

Watch the Youtube video for the game “Smooth Criminal”:
https://www.youtube.com/watch?v=3HhB6JzjtwE

Create a 2D top-down interactive stealth game.

The player character should have goals or target, such as hacking the computers or sealing an object. The NPCs in the game should patrol and guard the target from the player.

Each guard (or NPC) should have a ‘cone of sight’ which indicates whether the guard can ‘see’ the player. If the player ever enters a guard’s cone of sight, then the alarm is raised.

Once the alarm is activated, the players location is noted, and all guards will converge on the player attempting to catch (or shoot?) them. While the player remains visible to any one guard, all guards will know the location of the player.

If the player goes out of sight after raising the alarm, the guards should convincingly check the last known location of the player before resuming their patrol route.

The player must navigate through the game world, avoiding being seen or caught by the guards. Once they player reaches the goal (for example, hacks all the computers or opens the safe) they must reach the exit location to end the game.

Implementation:

Create a simulation with NPC guards that patrol your level. The guards should implement an appropriate NPC AI strategy.

When the player enters a guard’s area of visibility, the guard informs all other guards of the players location. Each guard should then form an appropriate strategy to apprehend the player. For example, you could have each guard acting individually, or have all guards attempt to catch the player by working as a group.

When the player is no longer visible to any guard, the guards should search for the player based on their last known location.

Pathfinding algorithms must be used for the NPCs. You may choose to implement a chess board-like series of cells for each guard to move to, or use each room in your map as a node in a pathfinding graph.

Optional Additions:

Feel free to expand on these base requirements and improve your simulation in any way you prefer.

Additional features you may wish to add could include:

- Blackboards:
  When a guard detects the player it writes a message to the blackboard. Any other guard nearby could read the message on the blackboard and be aware of the player, even if they are not able to detect it directly.
- Cameras:
  Add security cameras to your game that can also detect the player. The player may have the option of hacking cameras to disable them.
- Modified behaviours once detected:
  Have your guards change their behaviour once the player is detected. Perhaps they systematically search for the player or patrol a randomized route.
- Other Features:
  The video of the game "Smooth Criminal" demonstrates many additional features. For example, if the player kills a guard, other guards may discover the body and raise the alarm. Watch the video and see if you can add any of the features shown.

Submission:

You will need to submit the following:

- A Release build of your application that can execute as a stand-alone program
- Your complete Visual Studio project
  - Be sure to remove any temporary build folders (i.e., the Debug and Release folders). Only project files, source code files, and any resource files used should be included in your submission.

Package all files in a single compressed archive file (.zip, .7z, or .rar)

Submission Checklist:

This submission checklist is used to assist your assessor in marking your assessment.

A copy of this checklist can be downloaded from https://aie.instructure.com/. You must complete this checklist yourself and submit it with your project.

General

Description	**Y/N**
The program contains no syntax errors	
The program performs as described in the general description	
The program contains no logical errors	

The code is sufficiently commented and clean	
An attempt has been made to increase the program's efficiency	
Code compiles without errors or warnings	
Program executes without crashing	
Program has no memory leaks	
A release executable has been made and included in the submission	
Project files and source code are included in the submission	
All files are packaged in a single compressed archive	

Estimate the number of hours taken to complete this assessment	
How many times have you submitted this assessment (including this time)?	

Required Features

Complete the following table by providing the class name or file name, along with the line number, to show where you have implemented each feature.

Feature	**Class/File**	**Line Number**
The program implements a pathfinding algorithm		
The program implements an NPC AI strategy		

Feature	**Y/N**
Documentation discusses AI strategies for NPCs with reference to technical feasibility and responds to the design brief	
Documentation discusses a range of possible goals and actions, and other factors in the design of NPC AI	
Documentation includes images that effectively describe and communicate the design of implemented systems	
Post-mortem documentation discusses the difficulty of implementing planned techniques	
Post-mortem documentation reviews implemented techniques and discusses how they could have been improved	

Appendix 2

Technical Design Document Template

1.0 Revision History

<As you revise the document, list what was changed and when it was changed>

Version	Description
1.0	Initial document

2.0 Development Environment

2.1 Game Engine

<Proprietary/Unreal/Unity and version>

2.2 IDE

2.3 Source Control procedures

2.4 Third Party Libraries

2.5 Other Software

<2d art assets, audio, 3d modelling etc.>

3.0 Game Overview

3.1 Technical Goals

<3d graphics, 60fps, Challenging AI etc.>

3.2 Game Objects and Logic

<A list of logical elements in the game, i.e. door, button, pistol, ammo, light, bullet, wall, character etc. and description of their behaviour and purpose>

3.3 Game Flow

<description of what the player can do (actions) from the start menu to playing the game, through to hitting quit. Include how to win, how to lose, how the player is moved, and what programmer things might need to be considered>

4.0 Mechanics

<A list of the core game mechanics. I.e., what the player can do and how they achieve this, and what this triggers in the game. For example, shooting enemies is a core mechanic in an FPS>

5.0 Graphics

<Describe graphics features here. I.e., is your game top-down 2D? >

6.0 Artificial Intelligence

<Describe how AI works, i.e. state machine, fuzzy logic, GOAP. Describe the various behaviours and how they change behaviour, how do the 'creatures' in the game evaluate the world>
<include diagrams/flowcharts showing decision making processes>

7.0 Physics

<if needed>

<What engine are you using, what features from it (spring? Colliders?) how will physics be handled for objects? (box or sphere collider for objects, capsule for player) need to record specific locations for any reason? Potential slowdowns and how to mitigate.>

8.0 Items

<List of items you can pick up that can affect the player, and what they will affect, like 'picking up the hammer (refer collisions above) adds 5 to the players attack attribute'. Include details on how items influence gameplay or AI logic.>

Item	Parameter	Parameter	Parameter	Description
Default	6	6	6	
Weapon	5	7	7	
Weapon	8	5	5	
Weapon	5	Possible 10	NA	text
Weapon				

9.0 Game Flow

9.1 'Mission' / 'Level' structure

<Are all levels stored in memory? what data is saved across levels, are levels loaded synchronously to prevent pauses?>

9.2 Objectives

<What does the player try to accomplish on each level/mission? How is the players progress evaluated?>

10.0 Levels

<If any of the Levels require specific behaviours, describe those here>

11.0 Interface

11.1 Menu

<What are the menu options and what do they do?>

11.2 Camera

<Describe the camera, how it moves, perspective/orthographic, can it switch? How? Does it need to render-to-texture? does it prevent itself going through walls, use flowcharts to document behaviour>

11.3 Controls

<Keyboard, tablet touch/swipe/tilt, joystick, mouse etc. record double taps, multi touch, use mouse smoothing/ scale mouse for aiming etc.>

14.0 Asset List

<List all files needed, along with known attributes >

16.0 Technical Risks

<if you want your game to be a 1000 player pvp battle royale with 4k 120fps graphics, you need to say if this is doable and how you intend to do it>

