Outline

- Student Presentation: “Artificial Stupidity: The Art of Intentional Mistakes”
- Student Presentation: “Expressive AI: Games and Artificial Intelligence”

- Real-Time Strategy Games
  - Basic RTS AI
  - Advanced RTS AI
  - Starcraft AI Competition

- Project Discussion
Expressive AI: Games and Artificial Intelligence

- Presentation by: Ahsen Jaffer
- http://www.screenr.com/d6as
Outline

• Student Presentation:
  “Artificial Stupidity: The Art of Intentional Mistakes”
• Student Presentation:
  “Expressive AI: Games and Artificial Intelligence”

• Real-Time Strategy Games
  • Basic RTS AI
  • Advanced RTS AI
  • Starcraft AI Competition

• Project Discussion
Real-Time Strategy Games

• **Strategy games:**
  • Require the player to develop a plan to achieve certain goal (strategy)
  • E.g.: Chess
  • Turn-based or real-time

• **Real-Time Strategy (RTS) games:**
  • Player needs to develop and execute the plan in real-time: no time to stop and think with calm
Real-Time Strategy Games

- First RTS game ever: **Nether Earth** (ZX Spectrum, 1987)
  - [http://www.youtube.com/watch?v=OjQXAeB6zpU](http://www.youtube.com/watch?v=OjQXAeB6zpU)
Real-Time Strategy Games

• First “modern” RTS Game: **Dune II** (Commodore Amiga 1992)
  • [http://www.youtube.com/watch?v=KibE3B6syNQ](http://www.youtube.com/watch?v=KibE3B6syNQ)
Real-Time Strategy Games

- 3D RTS Games: **Homeworld** (PC 1999)
  - [http://www.youtube.com/watch?v=xQfuhxSlexI&feature=related](http://www.youtube.com/watch?v=xQfuhxSlexI&feature=related)
Real-Time Strategy Games

• Modern RTS Games: **Starcraft II** (PC/Mac 2010)
  • [http://www.youtube.com/watch?v=c3trTxsiKZI&feature=related](http://www.youtube.com/watch?v=c3trTxsiKZI&feature=related)
AI for RTS Games

- RTS Game AI:
  - Consumes most of the AI programmers resources
  - Amongst the most computationally expensive

- What needs to be controlled?
  - Military units
  - Economic units (workers)
  - Pathfinding
  - Town/base building
  - High-level Strategy
  - Tactical support systems (terrain analysis, opponent modeling, reconnaissance, etc.)
AI for RTS Games

• Why is it hard?
  • Many units to control
  • Incomplete information (fog-of-war)
  • Heavy emphasis on micro actions
  • Real-time

• The AI not only has to play well, it has to be fun:
  • Challenging enough for the player
  • Not frustrating
  • Fair
Outline

• Student Presentation: “Artificial Stupidity: The Art of Intentional Mistakes”
• Student Presentation: “Expressive AI: Games and Artificial Intelligence”

• Real-Time Strategy Games
• **Basic RTS AI**
• Advanced RTS AI
• Starcraft AI Competition

• Project Discussion
Basic RTS AI

• Building an AI to play an RTS game is complex.

• Although there is great variability in RTS games, AI-wise, they are quite similar.

• In this section we will learn how to create a very basic, simple and expandable architecture for RTS AI.

• The task can be much easier if a proper architecture is used:
  • Divide and conquer: different modules to handle different subproblems (pathfinding, construction, etc.)
Basic RTS AI Diagram

Perception
- Analyze Game State: Extract Useful Information

Strategy
- High-level Strategy: Army composition, priorities

Give Orders
- Middle level tasks: Base building, research, etc.

Execute Orders
- Auxiliary and Individual unit control: Pathfinding, Building placement, etc.
Basic RTS AI Diagram

Perception
- Unit Analysis
- Map Analysis

Strategy
- Strategy

Give Orders
- Economy
- Logistics
- Attack
- Arbiter

Execute Orders
- Unit AI
- Building Placer
- Pathfinder
Basic RTS AI Diagram

Perception
- Unit Analysis
- Map Analysis

Strategy
- Economy
- Logistics
- Attack

Give Orders
- Arbiter
- Unit AI
- Building Placer
- Pathfinder

Execute Orders

- Find paths between any two points A and B
- Typically 90% of the CPU
- Basic Algorithm: A*
Basic RTS AI Diagram

- Perception
  - Unit Analysis
  - Map Analysis

- Strategy

- Give Orders
  - Unit Analysis
  - Map Analysis
  - Economy
  - Attack
  - Building Placer
  - Pathfinder

- Build Orders

- Strategy
  - Finds locations for new buildings
  - Inputs: desired space, preferences (close to resources, close to enemy, etc.)
  - Typical algorithm: spiral search, or influence maps
Each type of unit in the game will have special code to execute its actions (movement, attack, etc.). Typically: Finite-state machine.
Basic RTS AI Diagram

- **Unit Analysis**
  - Perception
  - Strategy
  - Give Orders
  - Execute Orders

- **Economy**
  - Logistics
  - Attack

- **Arbiter**

- **Unit AI**
  - Building Placer
  - Pathfinder

- **Goal:** ensure there are enough resources
- **Input:** directives from strategy
- **Typical algorithm:** rule-based
Basic RTS AI Diagram

- **Goal:** handle dependencies between units, e.g.: creating unit A requires unit B. If Economy wants A, Logistics will detect that A is needed and create it.
- **Typical algorithm:** rule-based
Basic RTS AI Diagram

- Goal: destroy the enemy
- Given the directives of strategy, trains units, selects targets and sends units to attack
- Typical-algorithm: rule-based
Basic RTS AI Diagram

- Some games might require additional execution-layer modules: exploration, defense, research, diplomacy
- Attack module can be very simple or very complex
All the other modules want to execute actions. This module makes sure there are no conflicts (e.g. two modules wanting to control the same unit)

Typically modules generate requests with preferences, Arbiter chooses the order of execution
Basic RTS AI Diagram

- Sets the targets for all the lower modules (e.g. % of economy to be spent in attacking, types of units to use)
- Typical algorithm: finite-state machine, or rule-based
Basic RTS AI Diagram

- Prepares all the information about units the rest of modules might need (idea is to do it only once, so other modules do not need to be recomputing information again and again)
Basic RTS AI Diagram

- **Perception**
  - Unit Analysis
  - Map Analysis

- **Strategy**
  - Prepares all the information about the map other modules might need:
  - Buildable locations, choke points, resource locations, etc.

- **Give Orders**
  - Economics
  - Logistics

- **Execute Orders**
  - Unit AI
    - Building Placer
    - Pathfinder
  - Attack
Basic RTS AI Diagram (Project 1)

Perception
- Unit Analysis
- Map Analysis

Strategy
- Strategy

Give Orders
- Economy
- Logistics
- Attack
- Arbiter

Execute Orders
- Unit AI
  - Building Placer
  - Pathfinder
Example Basic RTS AI: Perception

• Most basic things to perceive:
  • Player data:
    • Number of units of each type available
    • List of units of each type that are idle
    • Action that each unit is executing
    • Available building locations
  • Enemy data:
    • Enemy positions seen (to know where to explore)
    • Enemy units seen (important to know the type)

• This module typically contains just custom code to compute all of the above
Example Basic RTS AI: Strategy

- Finite-State Machine

Resource spending:
- 80% Economy,
- 20% Military
- 2 workers wood
- 1 worker metal
- Army Composition: 100% footmen

After training 4 footmen

Resource spending:
- 20% Economy,
- 80% Military
- 2 workers wood
- 4 workers metal
- Army Composition: 100% knights

If enemy has Flying units

Army Composition:
- 50% knights
- 50% archers

If enemy has no more flying units
Example Basic RTS AI: Economy

- Given a set of targets (set by the strategy module):
  - Number of workers gathering certain resources
  - Number of resource buildings of certain type
  - Reaching a certain amount of supply
  - Etc.

- Collection of rules to achieve those targets:
  - E.g.: If workers gathering gold < target and idle workers > 0 then send an idle worker to gather gold.
  - A small set of rules is typically enough

- Output of this module are “requests” (to be sent to the logistics module):
  - Request: desired action, resources needed, priority
Basic RTS AI Diagram

Perception
- Unit Analysis
- Map Analysis

Strategy
- Economy
- Logistics
- Attack
- Arbiter

Give Orders
- Unit AI
- Building Placer
- Pathfinder

Execute Orders
Example Basic RTS AI: Attack

• Given a set of targets (set by the strategy module):
  • Army composition (e.g. 50% archers, 50% spearmen)
  • Wave size

• Collection of rules to achieve those targets:
  • E.g.: If available attack units > wave size then send wave to attack

• Typically decomposed in several subcomponents:
  • Army training
  • Target selection

• Output of this module are “requests” (to be sent to the logistics module):
  • Request: desired action, resources needed, priority
Example Basic RTS AI: Logistics

- Makes sure all dependencies are satisfied, e.g.:
  - If to train an “archer” you need a “lumber mill”, when the attack module sends the request to build an archer, it ensures there is a lumber mill first (otherwise, it creates a request to build one)

- It contains the graph of dependencies:
  - Which buildings/techs/units are required before we can execute certain actions

- Given the requests of the other modules, the proper dependencies are enforced.
- Logistics simply helps the other modules not having to worry about this. If the attack module wants an archer, it simply requests one, and this module does all what is necessary to get one

- Output are also requests, to be sent to the arbiter
Example Basic RTS AI: Arbiter

- **Given:**
  - The priorities set by the strategy module (% of resources to be spent in different areas)
  - The priority of each request (set by the individual modules)
  - The resources required by each request
- Ensures actions are executed only when they have enough resources, and that there are no conflicts (basically **sorting**)
- **E.g.:**
  - Request R1 has HIGH priority and needs 100 resources
  - Request R2 has LOW priority and needs 50 resources
  - Currently we have 80 resources
  - Arbiter ensures that we wait until 100 are collected, so R1 executes before R2

Request list
Example Basic RTS AI: Building Placer

- Simpler version: spiral search given the player starting point

Can be improved with preferences:
- Close to minerals
- Close to enemy base
- Away from enemy base
- Close to a choke-point
Basic RTS AI Diagram (Project 1)

Perception
- Unit Analysis
- Map Analysis

Strategy
- Economy
- Logistics
- Attack
- Arbiter

Give Orders
- Unit AI
- Building Placer
- Pathfinder

Execute Orders
Example in S3

• Result of implementing exactly the previous idea in S3
• Results in an AI that is:
  • Easy to implement
  • Plays good enough
  • Easy to expand

• Demonstration
Outline

• Student Presentation: “Artificial Stupidity: The Art of Intentional Mistakes”
• Student Presentation: “Expressive AI: Games and Artificial Intelligence”

• Real-Time Strategy Games
  • Basic RTS AI
  • Advanced RTS AI
  • Starcraft AI Competition

• Project Discussion
Advanced RTS AI

- Waypoints
- Influence Maps
- Hierarchical AI
- Squads
- Real-time Path Finding (week 3)
- Decision Making (week 4)
Waypoints

• Mark special locations in a map (used in all game genres)
Waypoints

- Most common usage: faster pathfinding
Waypoints in RTS Games

• Map annotations:
  • By hand (map editor)
  • Automatic (terrain analysis)

• Uses:
  • Path-finding (find a path through the way points)
  • Strategy/Tactics:
    • Mark Chokepoints
    • Resource-abundant spots

• Examples:
  • When building a defensive building (e.g. a tower), place it near the closest chokepoint
  • When building a new base, use the closest resource-abundant spot
  • Use waypoints in strategic decisions: “If a chokepoint separates the enemy from the player, defend the chokepoint”
Influence Maps

• Divide map in regions (e.g. grid):
  • Ideally, each region should share similar properties

• Store military influence:

Example:
Potential Field
Friendly troops have a positive influence
Enemy troops have a negative influence
Potential Field for Tactical Movement

- Coordinates where we can shoot the enemy are positive
- Coordinates where the enemy can hit us are negative
Potential Fields for Pathfinding

- Place a potential field that marks locations where the enemy can damage our units
- The cost of traversing each cell in the map is a function of how much damage the enemy can do to us in that cell
- Result:
  - A* would return paths that are a tradeoff between length and safety
Hierarchical AI

- Each layer defines a different level of abstraction for the level above
- Upper layers consider the layer below as “the game”
Squads

• Divide troops in squads:
  • Each squad has its own target
  • Typically composed of identical units
• Benefits:
  • Pathfinding: only one unit needs to pathfind, the other just follow
  • Formations (looks smarter and more realistic)
• Allows for more advanced strategies:
  • Some squads defend, while others attack
  • Attack multiple targets at once
  • Required for games with more than 2 players
Outline

• Student Presentation: “Artificial Stupidity: The Art of Intentional Mistakes”
• Student Presentation: “Expressive AI: Games and Artificial Intelligence”

• Real-Time Strategy Games
  • Basic RTS AI
  • Advanced RTS AI
  • Starcraft AI Competition

• Project Discussion
Starcraft AI Competition

- RTS AI is a big challenge, receiving a lot of interest
- Multiple competitions:
  - Starcraft
  - ORTS
- Independent competitions:
- Students doing the project in Starcraft can participate
Starcraft Man vs Machine

• The winner of the competition (Skynet) played against an expert human (Oriol Vinyals)

• [http://www.youtube.com/watch?v=7ffYOcXR7z8](http://www.youtube.com/watch?v=7ffYOcXR7z8)

• As of now, no AI can defeat the top human players
Outline

• Student Presentation:
  “Artificial Stupidity: The Art of Intentional Mistakes”
• Student Presentation:
  “Expressive AI: Games and Artificial Intelligence”

• Real-Time Strategy Games
  • Basic RTS AI
  • Advanced RTS AI
  • Starcraft AI Competition

• Project Discussion
Project 1: RTS Games

• Issues installing Starcraft / BWAPI?
• Issues running S3?
• What exactly should we do?