



## Raven: An Overview

### Artificial Intelligence for Interactive Media and Games

Professor Charles Rich  
Computer Science Department  
rich@wpi.edu

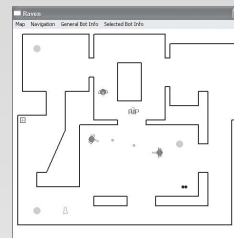
*[Based on Buckland, Chapter 7 and lecture by Robin Burke]*

CS/IMGD 4100 (B 16)

1

## Raven Game

- Quake-style death match
  - player and opponents (“bots”)
  - automatic respawning
  - rooms and corridors
  - weapons and health packs
- Top-down 2D simple rendering
- Player can “possess” a bot to play



## Familiar Techniques in Raven

---

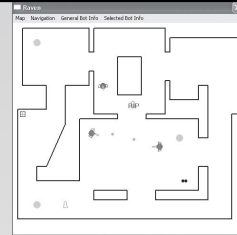
- Covered in IMGD 3000/4000 (or review on your own)
  - Steering (Chapter 3)
  - Path Planning (Chapter 5, 8)
- Studied this term
  - Messaging
  - (no state machines!)

## New Techniques in Raven

---

- Generic
  - triggers
  - sensory memory
  - **goal (behavior) trees**
  - fuzzy logic
- Genre-specific
  - targeting
  - weapon selection

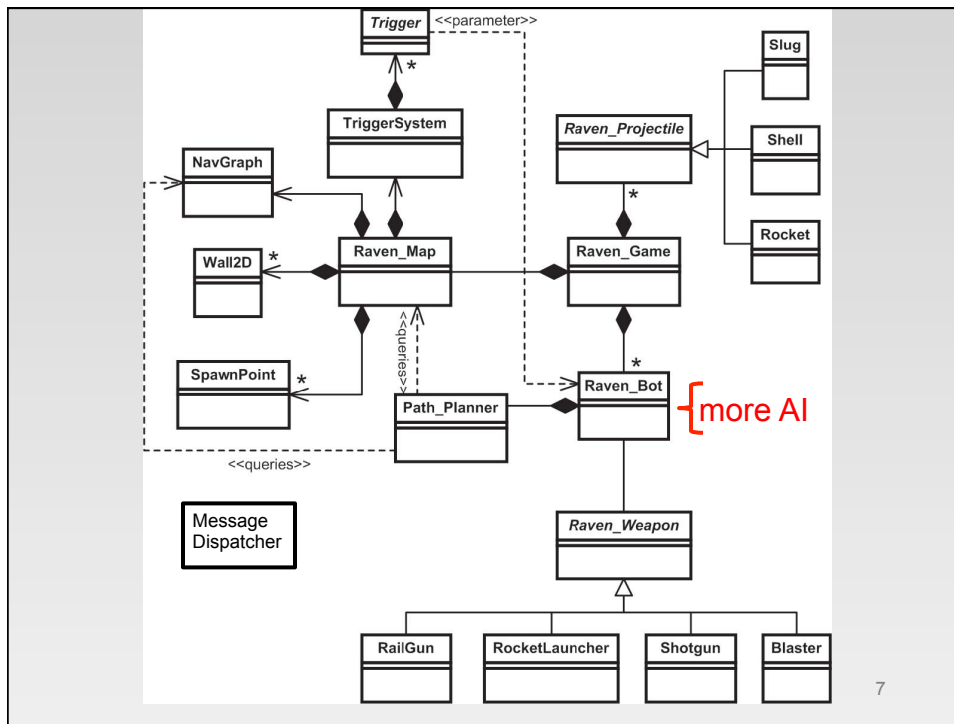
## Player Controls



- Right click to select bot
  - to observe internal state
- Possess selected bot by right clicking
  - left click to shoot
  - right click on map to navigate to point (AI path planning assist)
  - mouse position determines direction of firing
  - change weapons with 1, 2, 3, 4 keys
  - X key to release
- **Demo**

## Raven Schedule

	Tue, Nov 15	Chapter 7	Raven Anatomy	
	Wed, Nov 16			7 - Tournament Team [10%]
	Thu, Nov 17	Chapter 7	Raven Anatomy	
	Fri, Nov 18		<b>Soccer Tournament (HL 230 !)</b>	
	Sun, Nov 20			8 - My Bot [3%]
<b>5</b>	Mon, Nov 21		Natural Language and Dialog	
	Tue, Nov 22		Natural Language and Dialog	
	Wed, Nov 23		Thanksgiving Break	
<b>6</b>	Mon, Nov 28	Chapter 9	Goal-Driven Behavior	
	Tues, Nov 29	Chapter 9	Goal-Driven Behavior	
	Weds, Nov 30			9 - Steal Health [5%]
	Thu, Dec 1	Chapter 9	Goal-Driven Behavior	
	Fri, Dec 2		<b>Brainstorming:</b> Raven Bot Strategy	
	Sun, Dec 4			10 - Bot Design [3%]
<b>7</b>	Mon, Dec 5	Chapter 10	Fuzzy Logic	
	Tue, Dec 6		<b>Research:</b> Narrative Planning / Course Eval	
	Wed, Dec 7			11 - AI Middleware [10%]
	Thu, Dec 8		<b>Presentations:</b> AI Middleware	
	Fri, Dec 9		<b>Presentations:</b> AI Middleware	
	Sun, Dec 11			<b>(Due 6pm!)</b> 12 - Tournament Bot [10%]
<b>8</b>	Mon, Dec 12		<b>Raven Tournament (GH 012 !)</b>	



## Lua Configuration File

-----[[ General game parameters ]]-

--the number of bots the game instantiates

NumBots = 3

BotNames = {}

BotNames[1] = "RB\_Bot"  
 BotNames[2] = "Raven\_Bot"  
 BotNames[3] = "Raven\_Bot"

params.lua  
 <name>\_bot.lua

--- or even...

FrameRate = 2 \* UpdateRate

## Lua Configuration File

---

- Load the file as a Lua script

```
luaL_dofile(pLua, "Params.lua")
```

- Access the global variables as parameters

```
int numBots = (int) PopLuaNumber(pLua, "NumBots")
```

(no LuaBind, just Lua)

## The Raven Map

---

- holds all world geometry
  - walls
  - triggers
  - spawn points
- supports navigation (using sparse graph)
- load from file
- map editor provided
- **Demo**

## Raven Messages

---

- Combat related (bot -> bot)
  - Msg\_TakeThatMF
  - Msg\_YouGotMeYouSOB
  - Msg\_GunshotSound
- Path finding
  - Msg\_PathReady
  - Msg\_NoPathAvailable
- Misc
  - Msg\_UserHasRemovedBot
  - Msg\_OpenSesame

## Raven Weapon Properties (Lua Config File)

---

- Weapon Properties
  - DefaultNumRounds
  - MaxRoundsCarried
  - RateOfFire
  - IdealRange
- Projectile Properties
  - MaxSpeed
  - Mass
  - Max Force
  - Damage

## Raven Weapons

---

- **Blaster**
  - default, infinite ammo
  - 3 shots/sec, 1 unit damage
  
- **Shotgun**
  - 10 pellets spread out, 1 unit damage per pellet
  - good for short to medium range
  - 1 shot/sec

## Raven Weapons

---

- **Rocket Launcher**
  - slow, medium range weapon
  - 10 units damage within blast radius
  - 1.5 shots/sec
  
- **Railgun**
  - extremely fast slug, 10 units damage
  - ideal for long-range (sniping)
  - only stopped by walls
  - 1 shot/sec

## Weapon Desirability

---

- given distance to target
- uses fuzzy logic (discuss later)
- called by AI

## Projectiles

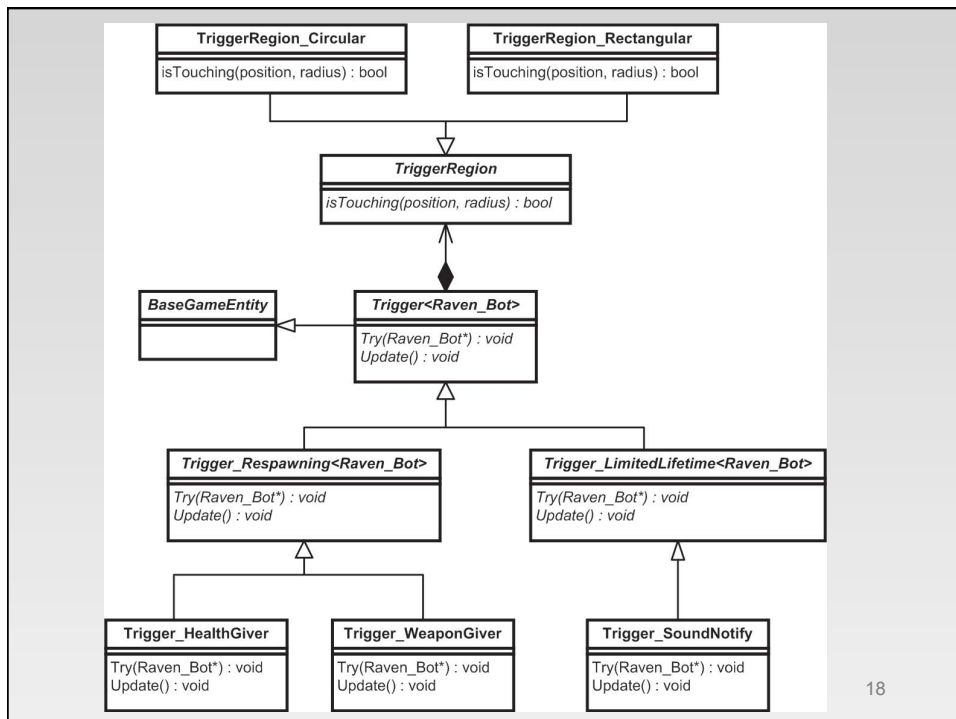
---

- point-mass physics simulation of flight using MovingEntity class
- when projectile intersects bot, it sends a [message](#)
  - who fired the shot
  - amount of damage done
- **Code walk**



## Triggers

- Very generic facility (typical in game engines)
- Trigger region
  - circle, rectangle in 2D
  - sphere, cube, cylinder, etc. in 3D
- When entity enters region (and perhaps other conditions checked), triggers arbitrary action
  - graphical
  - message passing
  - state change
  - etc.



## Respawning Triggers

---

- persistent, but becomes inactive for specified period of time after triggered
- *Weapon Giver*
  - calls PickupWeapon method on bot that triggers
- *Health Giver*
  - calls IncreaseHealth method on bot that triggers

## Limited Lifetime Triggers

---

- automatically removed after fixed number of update steps
- *Sound Notification*
  - new trigger created every time weapon fired
  - circular trigger region proportional to loudness
  - one tick lifetime
  - when triggered, sends message to triggering bot
    - identifies shooter

## Aside: Method vs. Message ?

---

- What's the difference between calling a *method* on the triggering bot (e.g., PickupWeapon) versus sending it a *message* e.g., Msg\_GunshotSound)?
  - *messages* handled based on recipient's current state (or goal)
    - one more level of indirection
    - easier to extend
  - *messages* have no return value (cf. return message)
  - *messages* support optional time delay
  - *messages* allow easier networked implementation
    - but still need to “serialize” parameters

## Managing Triggers

---

- TriggerSystem singleton
  - registers all triggers
  - updates triggers ('update' method)
  - renders active triggers
  - removes dead triggers
  - calls 'try' method on each active trigger

## Bot Intelligence

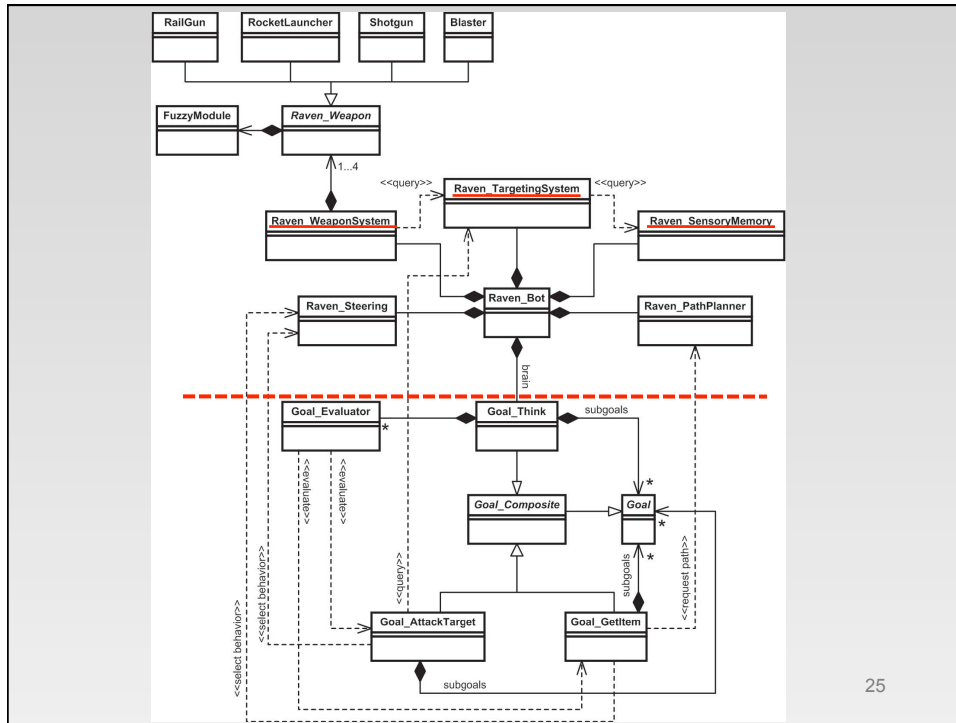
---

- *weapon handling* and *movement* operate totally independently!
  - at “lowest” level of AI, *always* choosing best target and weapon, aiming and shooting
  - i.e., heading (for aim) controlled separately from direction of movement
- higher level AI deals with “strategy”, which in this game means *where* to move (uses goal trees---see *next week*)

## Low-Level Bot Intelligence

---

- *Generic*
  - **Perception**
    - field of view
    - sensory memory
  - **Steering**
    - seek, arrive, wander, *wall avoidance*, *separation*
    - *no collision detection* or response between bots and world geometry
  - **Path Planning**
    - navigation grids provided by map
- *Genre-Specific*
  - **Target Selection**
  - **Weapon Handling**



25

## Modeling Perception

- can make a big difference in playability of certain types of games
- NPC's do not *inherently* have perceptual systems
  - can theoretically know everything about game state
  - *sometimes* this knowledge is needed to compensate for their (otherwise) stupidity
- But
  - designer must be very judicious
  - *too much* or *too little* perceptual knowledge can be bad

## Too Much Perceptual Knowledge

- seeing through walls
- eyes in the back of his head
- seeing in the dark (w/o night goggles)

*“perceptual omniscience”*

## Too Little Perceptual Knowledge

- deaf (e.g., to huge explosion nearby)
- blind (to unexpected objects)
- out of sight / out of mind

*“selective sensory nescience”*

## Perceptual Modeling in Raven

- **Field of View**
  - 180 degrees
  - cannot see other bots through walls
    - requires expensive calculation
  - “knows” the location of health packs and weapons
    - unfair?
- **Audio Triggers**
  - omni-directional
  - distance-limited
- **Sensory Memory**

## Sensory Memory Records

- **For each opponent (as encountered)**
  - most recent sensory event (seen or heard)
    - time
    - position
  - when first became visible (seen)
  - when last visible (seen)
  - within field of view?
  - shootable? (no obstructions)
- **Sensory memory updates list of such records**
  - GetListOfRecentlySensedOpponents

## Target Selection

---

- each bot has its own `TargetingSystem` instance
- current target updated regularly (param)
  - may be null
- Raven bots targeting criterion:
  - closest opponent
- other ideas:
  - weakest opponent in range
  - opponent that is shooting me
  - restrict to field of view (unless hear?)
  - etc...



## Weapon Handling

---

- each bot has its own `WeaponSystem` instance
- currently selected weapon and inventory
  - max one weapon of each type in inventory
- performance “imperfections”
  - reaction time
  - aiming accuracy
- aiming persistence (after opponent disappears)
- key methods
  - `SelectWeapon`
  - `TakeAimAndShoot`






## Update Frequency

---

- cannot update all AI components all the time
  - too expensive
  - not necessary
- Steering: every update (don't run into walls)
- Weapon selection: 2 Hz
- Goal arbitration (next week): 2 Hz
- Sensory memory update: 4 Hz  
(uses Regulator class)

## Coming up...

---

- **Fri (tomorrow)** Soccer Tournament in **HL 230**
  - **Need some volunteers at 9:30?**
- **Sun** midnight: "My Bot" homework due
  - copy and rename into folder (as before)
  - make small change in targeting
    - add special check for target within range that can kill with single firing of currently selected weapon
-  Thanksgiving!
- **Weds, Nov 30** midnight: "Steal Health" homework due
- **Fri Dec 2** Raven Bot Strategy Brainstorming  
[Raven Code Walk]