



Raven: An Overview

Artificial Intelligence for Interactive Media and Games

Professor Charles Rich
Computer Science Department
rich@wpi.edu

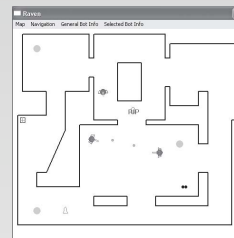
[Based on Buckland, Chapter 7 and lecture by Robin Burke]

IMGD 400X (B 08)

1

Raven Game

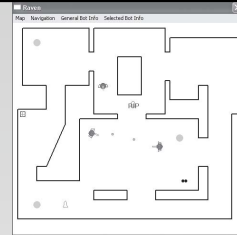
- Quake-style death match
 - player and opponents (“bots”)
 - automatic respawning
 - rooms and corridors
 - weapons and health packs
- Top-down 2D simple rendering
- Player can “possess” a bot to play



IMGD 400X (B 08)

2

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**

Old Techniques in Raven

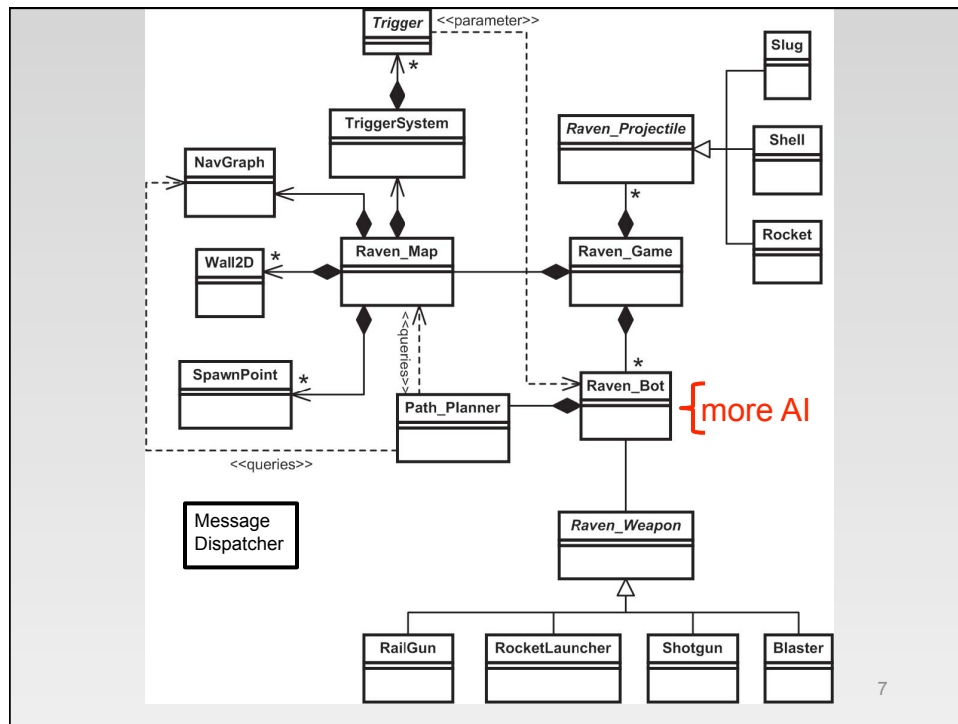
- Review from IMGD 4000
 - Steering (Chapter 3)
 - Path Planning (Chapter 5, 8)
- Studied this term
 - Messaging
 - (no state machines!)

New Techniques in Raven

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

Raven Schedule

4	Mon, Nov 17	Chapter 7	Raven Anatomy	
	Tue, Nov 18	Chapter 7	Raven Anatomy	
	Wed, Nov 19			7 - Tournament Team [10%]
	Thu, Nov 20	Chapter 7	Raven Anatomy	
	Fri, Nov 21		Soccer Tournament (IMGD Lab)	
	Sun, Nov 23			8 - My Bot [3%]
5	Mon, Nov 24		Futures: TBD	
	Tue, Nov 25		Futures: TBD	
	Wed, Nov 26		<i>Thanksgiving Break</i>	
6	Mon, Dec 1	Chapter 9	Goal-Driven Behavior	
	Tues, Dec 2	Chapter 9	Goal-Driven Behavior	
	Weds, Dec 3			9- Steal Health [5%]
	Thu, Dec 4	Chapter 9	Goal-Driven Behavior	
	Fri, Dec 5		Brainstorming: Raven Bot Strategy	
	Sun, Dec 7			10 - Bot Design [3%]
7	Mon, Dec 8	Chapter 10	Fuzzy Logic	
	Tue, Dec 9	Chapter 10	Fuzzy Logic	



Raven 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"

--- 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

```
numBots = 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 (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
 - 5 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

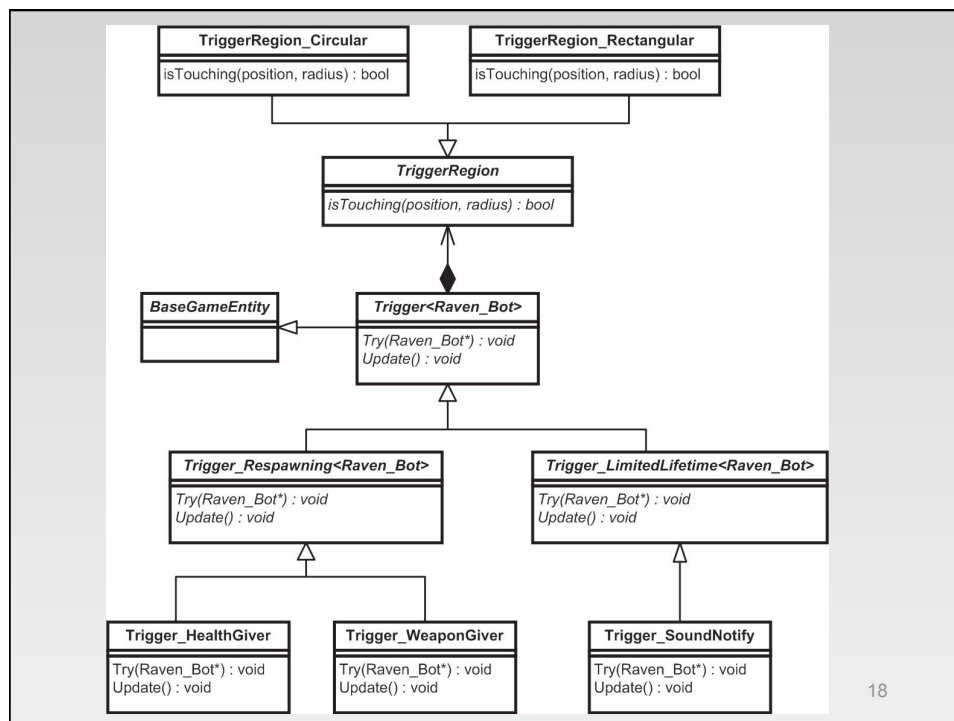
- given distance to target
- uses fuzzy logic (discuss later)
- called by AI

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

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
 - supports optional time delay
 - lends itself better to networked implementation
 - but still need to “serialize” parameters

Managing Triggers

- TriggerSystem singleton
 - registers all triggers
 - updates triggers
 - renders active triggers
 - removes dead triggers
- Code Walk

Bot Intelligence

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

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**

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 time 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
- Sensory memory update: 4 Hz
- Goal arbitration (next week): 2 Hz
- note Regulator class

Upcoming Events

- **Weds** midnight: Soccer Tournament Team due
- **Thurs** "Futures" lecture on AI and Animation
- **Fri** Soccer Tournament in IMGD Lab
- **Sunday** 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
- **Raven Code Walk**