BEHAVIOR TREES
Damián Isla,
Moonshot Games
Agenda

- Why FSMs Suck
- Behavior Trees v0
- Behavior Trees, v1
- Behavior Trees, v2+
- Why Behavior Trees Suck
- Conclusions
Warning

- "Behavior Trees" are a loose concept
- Terminology is far from standard
- Feature-set is far from standard
- This is going to be idiosyncratic

- Also Behavior Trees are Stupid
  - think of them as a convenient way to organize your thousands of "if-then-else" statements.
Let’s Situate Ourselves

SENSE

PERCEPTION

KR

PLAN

ACT

AIMING

ANIMATION

LOOKING

DIALOGUE
Why FSMs are Awesome

- Easy to understand
- Easy to implement
- Easy to debug
- VERY flexible
Inside a State

- Where I go
- How I get there
- What I look/aim/shoot at
- Special Animations
- whatever

- When I stop doing this and do something else instead
Why FSMs Suck
Why FSMs Suck
Why FSMs Suck
Why FSMs Suck

This is where the code lives

So code complexity grows $\sim N^2$
HFSMs

Still $N^2$, but at least $N$ is smaller
(Also, god forbid you decide to restructure this)
The Scalability Problem

Every time I want to add a new state, I have to think

- Where can this state come from?
- Where can this state go to?
- What am I breaking?
- And btw, what does this state actually do?
- And then I have to maintain all of this
The Scalability Problem

“Do $A$ whenever the player gets closer than 3 meters.”

- Duplicate this logic in each transition?
- What about when that logic changes?
- Every state needs to know a little about $A$
BEHAVIOR TREES V0
BTs Circa 1998

Children Compete!

- Child returns a floating-point desire-to-run
- Parent chooses highest-desire child
- + bonus for currently running child

Word of the day: Hysteresis
Inside a Behavior

• How much do I desire to run?
• Where I should stand
• How I get there
• What I look/aim/shoot at
• Special Animations
• whatever
• Children

WE JUST KILLED N^2!
Or Did We?
Or Did We?
Or Did We?

Charge:
  if (target closer than 3)
    desire = 5.2
  if (target facing me)
    desire += 1

Flee:
  if (target closer than 4)
    desire = 3.3
  if (target is player)
    desire += 2
Or Did We?

Charge:
  if (target shield down)
    return 1e5
  if (target closer than 3)
    desire = 5.5
  if (target facing me)
    desire += 1

Flee:
  if (target has rocket launcher)
    return 1e6
  if (target closer than 4)
    desire = 3.3
  if (target is player)
    desire += 2
Or Did We?

Floating-point desire-to-run implies some normalized scale of “desire” across all behaviors

- In practice, nearly impossible
- So we’re really back to $N^2$
  - for any scale > trivial

Note: If this were reinforcement learning, this would be $V^*(s)$ (total future discounted reward)
Insight #1

This is an instance of the Apples-to-Oranges problem.

How do you compare:

- desire to run away from a scary enemy
- desire to attack an enemy
- desire to take cover
- desire to jump in a friend’s vehicle
Insight #2

Designers (and players) think in terms of:

- triggers
  - not “exit conditions” (like FSMs)
- priorities
  - not squishy analog desires

- Run away from scary enemy
- Take cover
- Help friendly vehicle
- Attack enemy
BEHAVIOR TREES V1
HALO 2

The Big Goal: Scalability

Scalability?

No! Scalability!
Binary Relevancy

It is way easier to write a binary relevancy function than a floating point desire-to-run function.

Decision Policies:
- prioritized list
- sequential
- sequential-looping
- random
- etc.
Inside a Behavior

• Do I want to run right now?

• If I am running, do I want to continue running?

• Where I should stand
• How I get there
  ...

• Children

HYSTERESIS!
NOTE, BEHAVIORS ARE OFTEN DEACTIVATED THROUGH INTERRUPTION, RATHER THAN THROUGH EXPLICIT ENDING CONDITIONS
Problem: Non-constant Priority

UNLESS PLAYER IS IN VEHICLE, IN WHICH CASE...

- Charge
- Fight
- Enter vehicle

- Enter vehicle
- Charge
- Fight
BTv1 Feature 2

Impulses

- Enter player vehicle
- Charge
- Fight
- Enter vehicle

Alternative trigger conditions and priority for behaviors
Spore

Impulses became universal

- Enter player vehicle
- enemy nearby
- enemy present
- vehicle nearby

- Enter vehicle
- Charge
- Fight
- Enter vehicle
And lots more...

Tree masks

- super-fast condition bitvector for super common condition checks

ON FOOT

PASSENGER

DRIVER
And lots more...

Per-character tree customizations
And lots more...

Dynamic stimulus-driven branches
BTv1 Summary

- Binary Relevancy
- Per-parent customized decision routine
- Impulse-triggers for non-constant priorities

On the debugging Front

- Pretty easy to expose why something happened
- Figuring out why something DIDN’T happen is fiendishly hard.
BEHAVIOR TREES V2+
Behavior Trees v2

- Behavior trees are now VERY widely used

- Lots of enthusiasm for BTs as a broadly expressive language for crafting behavior
Behave 2 Unity Plug-in by Angry Ant
Behavior Trees as a Visual Language

- Just like, say, Unreal’s Kismet
- What is good about that?
- What is bad?
Behavior Trees as a Visual Language

Just like, say, Unreal's Kismet

What is good about that?

What is bad?
Lots of Potential Features

- Concurrent branches

Enter player vehicle

Parallel

Say “I’m gettin’ in!”
Enter vehicle

Sequence

Run to vehicle
Get in vehicle
Lots of Potential Features

- Concurrent branches
- Behavior parameters vs. Behavior instance parameters
Lots of Potential Features

- Concurrent branches
- Behavior parameters vs. Behavior instance parameters
- Decorators

Parallel Timer: 5s

- Enter player vehicle
- Timer: 5s
- Say “I’m gettin’ in!”
- Enter vehicle
Considerations

- How often are decisions made?
- Pre-validation or not?
  - this starts to get really close to an HTN
  - though HTNs are generally looking for a single solution
- How granular is the tree?
- Who makes the trees?
  - this is always more complicated than it looks
WHY BEHAVIOR TREES SUCK
Search Fragment

Presearch

Search

- Uncover
- Suppress
- Guard

- Uncover
- Search-sync
- Approach
- Pick next search point
void s_agent::behavior_update()
{
    if (omap.confusion < k_confusion_threshold)
    {
        s_pos2d pos;
        pos= omap.get_target_position();
        move_to(pos);
    }
}
void s_agent::behavior_update()
{
    if (omap.confusion < k_confusion_threshold)
    {
        s_pos2d pos;
        pos = omap.get_target_position();
        move_to(pos);
    }
}
CONCLUSIONS
Transparent Decision-Making

Players understand TRIGGERS, not TRANSITIONS
  - i.e. when the brute’s armor depletes, he charges, no matter WHAT he was doing at the time

Players understand priorities

Players understand discrete events

Players understand task-decomposition

→ The Tree is representative of the mapping we HOPE exists in the player’s head
AI in Halo 2 was known to be fun to play against and to make smart decisions, like flanking the player. Like in most games, this AI is fully scripted and contains no 'real' intelligence.
Thanks!