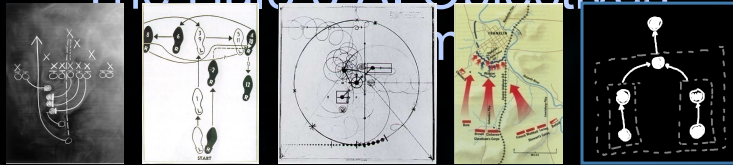


Building a Better Battle

The Halo 3 AI Objectives



Damián Isla
Bungie Studios



NON FACETE NOBIS CALCITRARE VESTRVM

Building A Better Battle

Designer tools

AI is an integral part of it

An interesting *Next-Gen* problem



NON FACETE NOBIS CALCITRARE VESTRVM



“Big Battle” Technology

- Precombat
- Scalable perception
- Effects
- In-game cinematics
- Scalable AI
- Combat dialogue
- Encounter logic
- Mission dialogue
- Ambient sound
- Flocking
- Targeting groups

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

“Big Battle” Technology

Activities

Scalable perception

Effects

Scalable AI

In-game cinematics

Combat dialogue

Encounter logic

Ambient sound

Flocking

Targeting groups

Mission dialogue

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

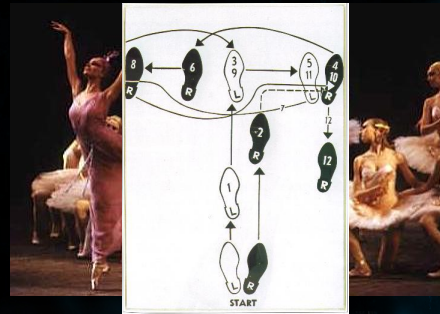


Encounter Design

- Encounters are *systems*
- Lots of guys
- Lots of things to do
- The system reacts in interesting ways
- The system collapses in interesting ways

An encounter is a complicated dance with lots of dancers

How is this dance choreographed?



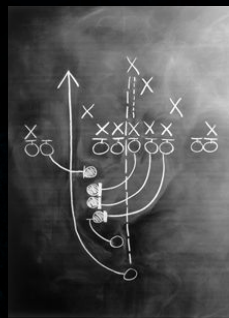
BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

Choreography 101

- The dance is about the illusion of strategic intelligence
- Strategy is environment- story- and pacing-dependent

Designer provides the strategic intelligence



AI acts smart within the confines of the plan provided by the designer

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

The Canonical Encounter

Two-stage fallback

- Enemies occupy a territory
- Pushed to “*fallback*” point
- Pushed to “*last-stand*” point
- Player “breaks” them
- Player finishes them off

... plus a little “spice”

- snipers
- turrets
- dropships



BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

Task

The *mission designers'* language for telling the AI what it should be doing

Halo:

- Territory
- Behavior
 - aggressiveness
 - rules of engagement
 - player following

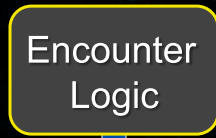


Changing task moves AI around the encounter space

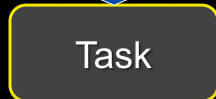
BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

The Control Stack



Mission-designers script sequence of tasks

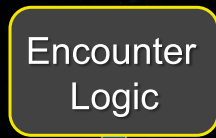


Mission designers



AI engineers, AI designers
Within the task, the AI behaves autonomously

The Control Stack



Mission-designers script sequence of tasks



~~within the task, the AI behaves autonomously~~

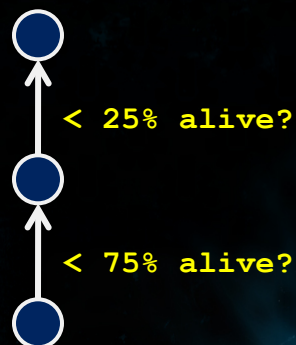
Halo 2: The Imperative Method

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

The Imperative Method

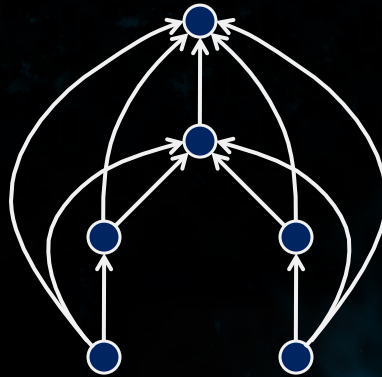
Give the designers an FSM construction tool



BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

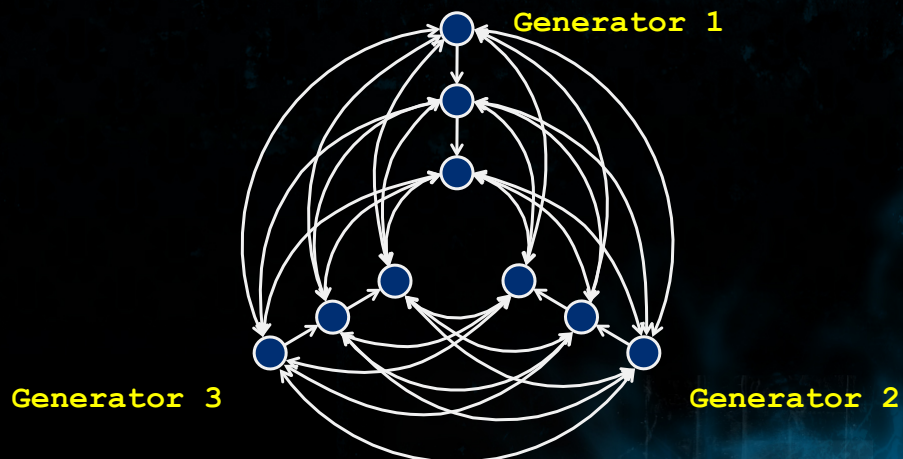
Problems with the Imperative Method



BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

Problems with the Imperative Method



Explicit transitions $\rightarrow n^2$ complexity

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

Problems with the Imperative Method

For Halo 3:

- Larger encounters
- More characters
- More open spaces
- More avenues of attack

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

Halo 3: The Declarative Method

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

The Declarative Method

The new approach:

Enumerate “tasks that need doing” in the environment

Let the system figure out who should perform them

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

The Declarative Method

Not without precedent



Similar to “affordances”

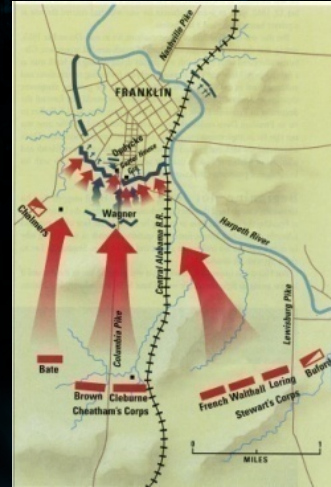
BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

The Declarative Method

Tasks have *structure*

- Relative priorities
 - “The *most important* thing is to guard the door, but if you can, also guard the hallway”
- Are made up of sub-tasks
 - “Guarding the hallway means guarding the front, the middle and the rear of the hallway.”



BUNGIE

NON FACITE NOBIS CALCITRARE VESTRUM

Behavior Trees

(Handling Complexity in the Halo 2 AI, GDC 2005)

Takeaways:

1. Prioritized-list decision scheme
2. Behaviors are self-describing

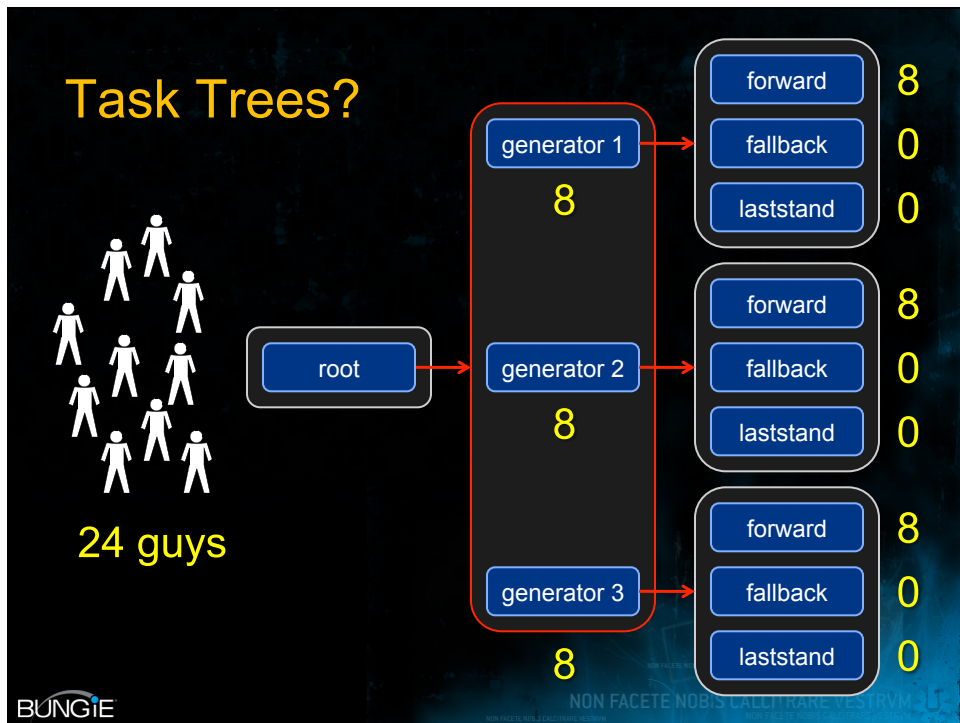
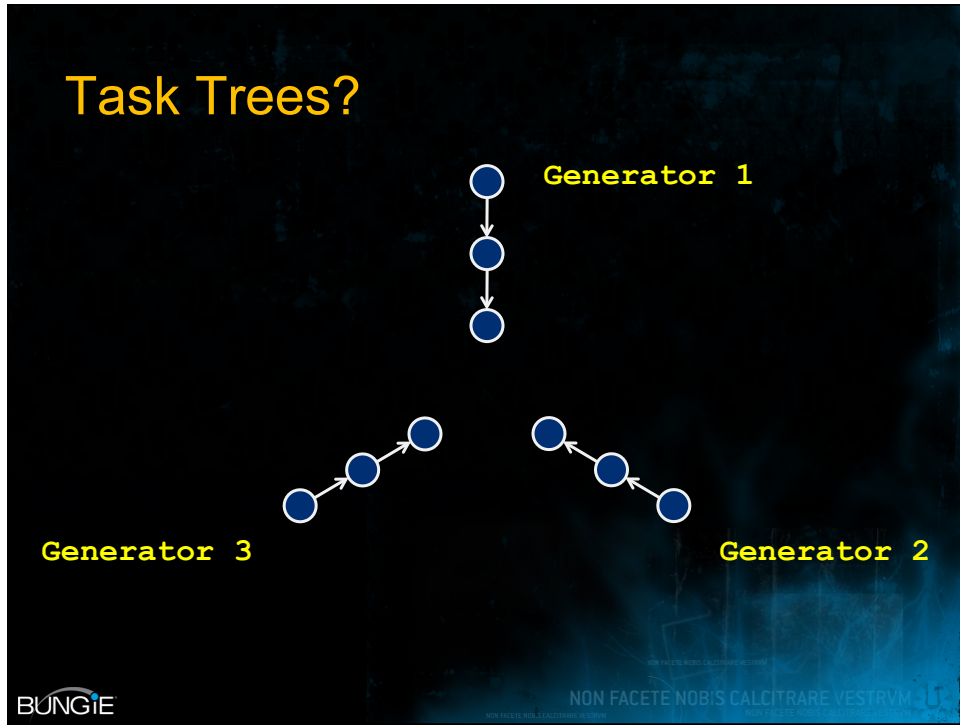


We are not making a *single* choice.

We are finding a *distribution* across *all* choices.

BUNGIE

NON FACITE NOBIS CALCITRARE VESTRUM



Halo 3 AI Objectives System

The structure:

- A Tree of Prioritized *Tasks*
- Tasks are self-describing
 - priority
 - activation script-fragments
 - capacities

The Algorithm:

- Pour squads in at the top
- Allow them to filter down to the most important tasks to be filling **RIGHT NOW**



Basically, it's a plinko machine.

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

The *Dynamic* Plinko Machine

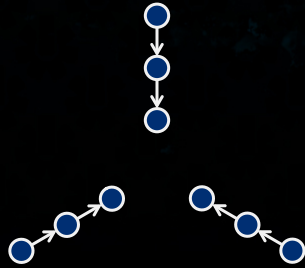
- Tasks turn themselves on and off
- Squads pulled UP, on activation of a higher-priority task
- Squads pushed DOWN, on deactivation of the task they're in



BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

3 Generators Revisited



NON FACETE NOBIS CALCITRARE VESTRVM

Designer UI

AI Objectives

Name: obi_ss_covenant Add Delete Render Firing Points

Zone: zn_substation

Task	Conditions	Filter	Style	Min	Max	Bodes	Life	Min Str	#fps
[0] phantom		phantom	Normal	0	0	0/0	0/0	0.00	3
[0] infantry_gate		none	Normal	0	0	0/0	0/0	0.00	0
[0] back_jackal_gate		jackal	Normal	0	0	0/0	0/0	0.00	0
[0] dock_gate	((= g_ss_obj_control 4))	none	Normal	0	0	0/17	0/0	0.00	0
[0] back_gate		none	Normal	0	0	0/0	0/0	0.00	0
[0] b_cov_back	(b= g_ss_obj_control 9)	leader	Normal	3	5	0/0	0/0	0.00	34
[0] b_front_01b	(and (not (volume_test_players tv_ss_07)) ((= g_ss_obj_control 7))	leader	Normal	0	15	0/4	0/0	0.00	70
[0] b_front_01a		none	Normal	0	0	0/2	0/0	0.00	161
[0] b_cov_03		leader	Normal	0	4	0/5	0/0	0.00	44
[0] b_cov_01	((= g_ss_obj_control 7))	leader	Normal	0	4	0/4	0/0	0.00	71
[0] b_cov_02	((= g_ss_obj_control 8))	leader	Normal	0	4	0/4	0/0	0.00	64
[0] brute		brute	Normal	0	2	0/3	0/0	0.00	64
[0] b_grunt_01	((= g_ss_obj_control 7))	grunt	Normal	0	3	0/0	0/0	0.00	47
[0] b_grunt_02	((= g_ss_obj_control 8))	grunt	Normal	0	3	0/0	0/0	0.00	46
[0] wayback		none	Normal	0	0	0/0	0/0	0.00	15

- Integration with HaloScript
- Run-time feedback



NON FACETE NOBIS CALCITRARE VESTRVM

The Algorithm

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

The Algorithm

- Consider a subtree fragment
- Determine which children are active
 - Squads in inactive tasks assigned back up to parent
- Consider top priority group
- Collect squads to attempt to distribute
 - Squads currently in parent
 - Squads in lower-priority tasks
- Distribute Squads
- Recurse for children in top priority-group
- Iterate to next “priority group”



BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

Squad Distribution

Formally, we have

- set S of n squads
- set T of m tasks

Now, find a mapping $F(S) \rightarrow T$

Two parts:

1. Respect Task-Capacity Constraints
2. Minimize cost function $H(F)$

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

Squad Distribution

1. Respect Task-Capacity Constraints

guys assigned to task $t \leq \text{capacity}(t)$

... but remember, we're bucketing by squads.



BUNGIE

This is called **bin-packing**. And it's NP-Hard.

NON FACETE NOBIS CALCITRARE VESTRVM

Squad Distribution

1. Respect Task-Capacity Constraints

Fortunately

- a) there's always Wikipedia
- b) we can live with sub-optimal
- c) we're optimizing not for m , but for $H(F)$

BUNGIE

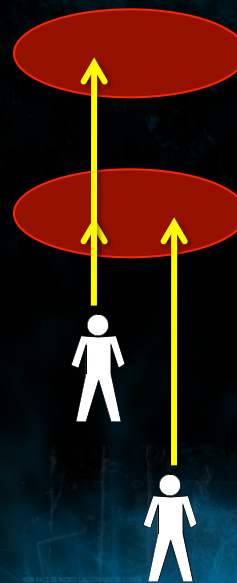
NON FACETE NOBIS CALCITRARE VESTRVM

Squad Distribution

2. Minimize cost function $H(F)$

Why a cost function?

- Gives us a basis for choosing one distribution over another
- Weigh different concerns
 - *don't want* to travel far
 - *want* to act coordinated
 - *want* to balance the tree
 - *want* to get near to the



BUNGIE

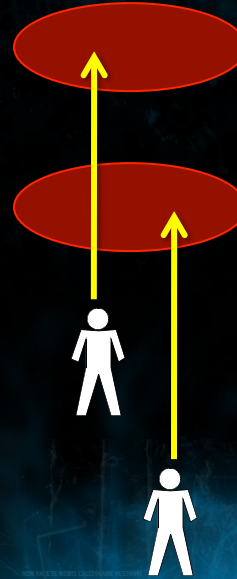
NON FACETE NOBIS CALCITRARE VESTRVM

Squad Distribution

2. Minimize cost function $H(F)$

DANGER: AI can look really stupid with wrong $H(f)$

OPPORTUNITY: Designer has abdicated his decision-making authority



BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

Squad Distribution

2. Minimize cost function $H(F)$

A class of cost functions:

$$H(F) = \sum_s H(s, F(s))$$

We use

$$H(F) = \sum_s \text{distance}(s, F(s))$$

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

A Greedy Approach

```

while (S is not empty)

    find pair (s,t) that give the minimum H
    (s,t) for all S x T (where adding s to t
    would not exceed t's capacity)

    if (s,t)
        assign(s, t)
        capacity(t) = capacity(t) - size(s)
        S = S - s
    else
        end
  
```

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

A note on Perf

Our algorithm may be $O(n^2m)$, but we are redeemed by the fact that n and m are small

Other perf measures

- Cache $H(s,t)$ results
- Timeslice entire trees ← Halo3
- Timeslice nodes within trees

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

Refinements

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

Filters

Particular tasks only available to particular *kinds* of guys

E.g.

- Must be of character type X
- Must be in vehicles
- Must NOT be in vehicles
- Snipers

“Filters”

- Specify *occupation* conditions (as opposed to *activation* conditions)
- “Trivially” implemented as an inf return value from $H(s, t)$
- Helpful for the “spice”

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

Further Task Refinements

Activation behavior

- Latch on
- Latch off / exhaustion

Exhaustion behavior

- Death count
- Living count

Assignment behavior

- One-time assignment

All of these were designer requests



NON FACETE NOBIS CALCITRARE VESTRVM

Case Studies



NON FACETE NOBIS CALCITRARE VESTRVM

Case Study #1: Leadership

Want to have leaders and followers

- Brute and three grunts
- Brute Chieftan and brute pack

Gameplay

- Leaders provide structure to encounter
- Leader death “breaks” followers

BUNGIE



Case Study #1: Leadership

Two Parts:

1. Leadership-based filters
 - Core task: “leader” filter
 - Peripheral tasks: “NO leader” filter
3. Task “broken” state
 - Task does not allow redistribution in or out while broken
 - NPCs have “broken” behaviors

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

Case Study #2: Player pickup

Vehicle encounters are not fun without a vehicle

Gameplay

- When the player needs a vehicle, allies go pick him up



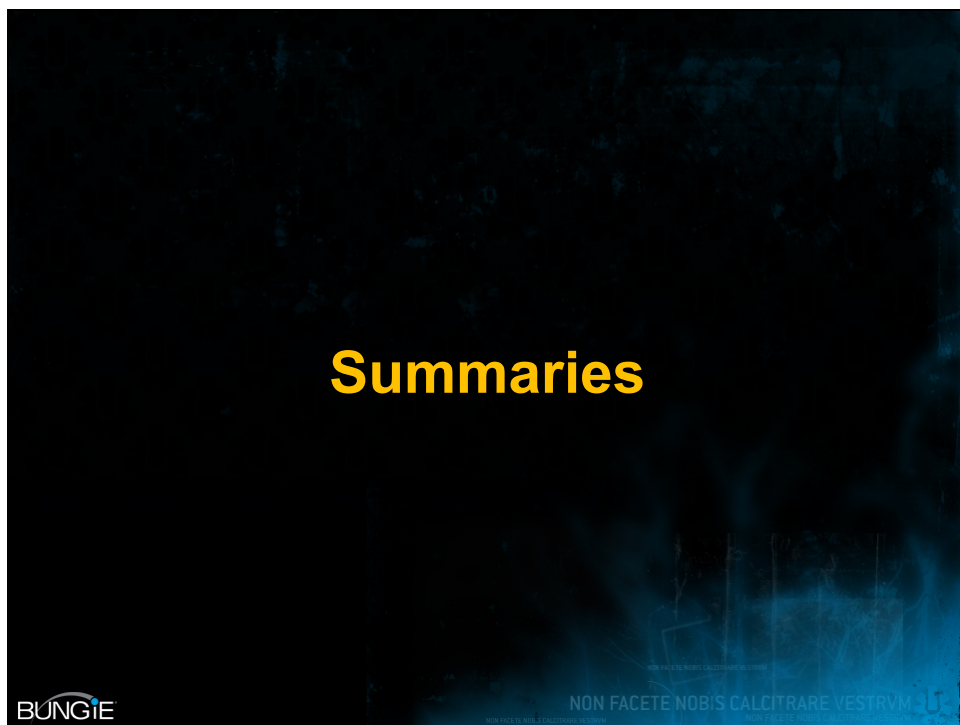
Case Study #2: Player pickup

Implementation: one dedicated player-pickup task per encounter

Four parts:

1. vehicle filter
2. `player_needs_vehicle()` script function
3. “follow player” task option
4. driver `player_pickup` behavior

And that's it!



Badness Summary

- Requires designer training
- Sometimes awkward relationship between scripting system and Objectives
- Tying together allied and enemy “fronts” was complicated.
- The squad wasn’t always the best level at which to do the bucketing
 - e.g. give a guy a sniper rifle ... shouldn’t he then be allowed to occupy a “sniper” task?

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

Technique Summary

- Declarative approaches are great
 - less direct control, more manageability
- Hierarchies are great
 - more modular
 - better scalability
- Self-describing tasks makes this whole thing $O(n)$ complexity rather than $O(n^2)$ (conceptually)

BUNGIE

NON FACETE NOBIS CALCITRARE VESTRVM

Production Summary

- The Goal: provide a powerful tool for designers to control strategy-level decision-making for a large group of characters
- Flexible enough to incorporate plenty of designer-requested features / modifications
- Great for Prototyping
 - became much more complicated as we neared shippable encounter state
- One-stop-shop for encounter construction



NON FACETE NOBIS CALCITRARE VESTRVM

Summary Summary

Not a problem isolated to Halo

As number of NPCs grows, these kinds of techniques will become more and more important

All you need ...

... is $H(s,t)$



NON FACETE NOBIS CALCITRARE VESTRVM