Modular AI Systems

Troy Humphreys
Lead AI Programmer
Turtle Rock Studios

Why this talk…

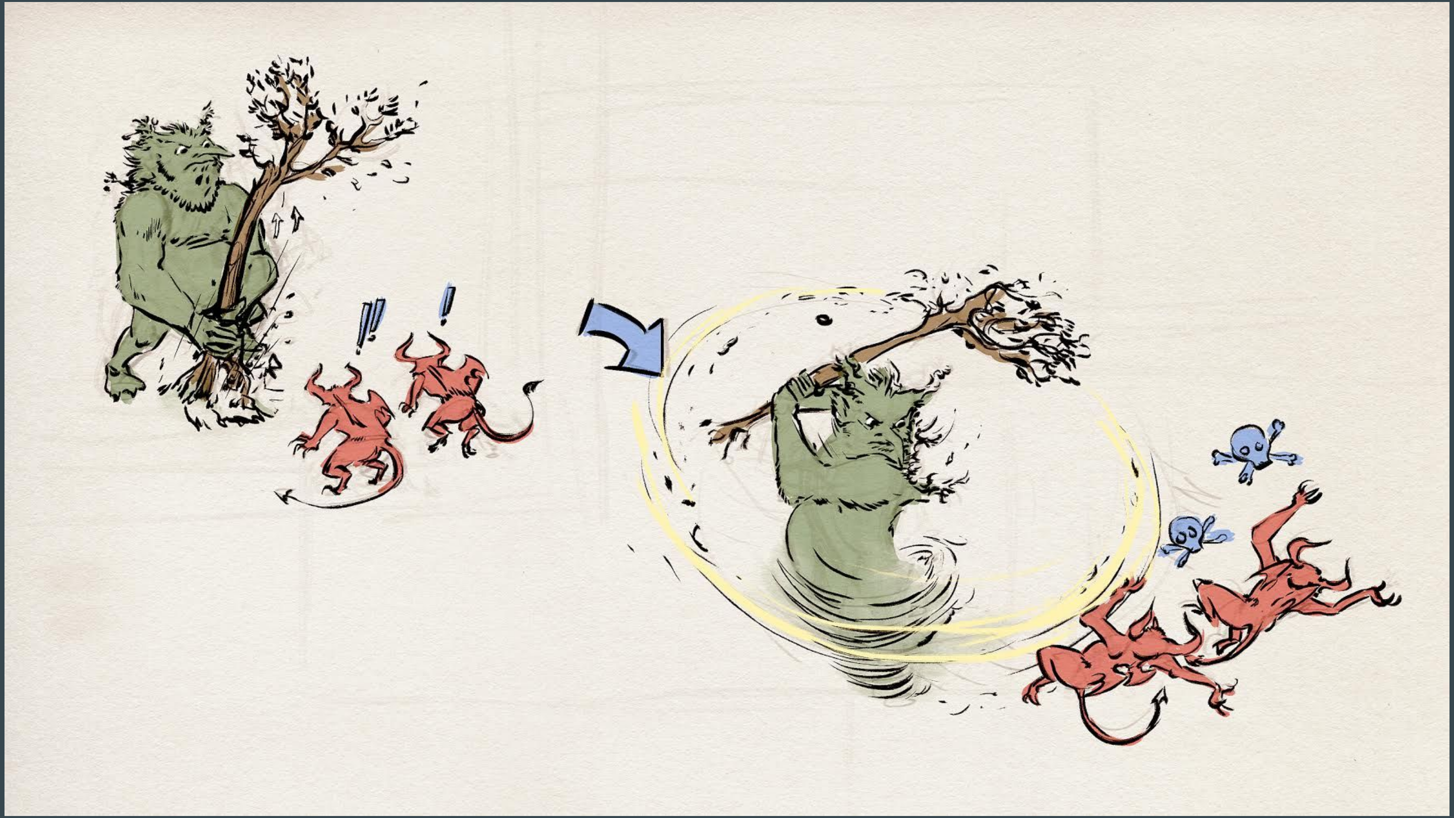
- Game AI architectures are pretty modular
- We used a behavior tree for the AI in Evolve
    - But the AI was still hard to get out the door
- So we set out to fix our AI system
    - Figure out why our modular system wasn't "modular enough" and fix it
    - Without breaking our legacy characters and the systems that run them

How do you fail at Modular?

•••

Let’s start with a simple example

Forest Troll - Tree Trunk Tornado

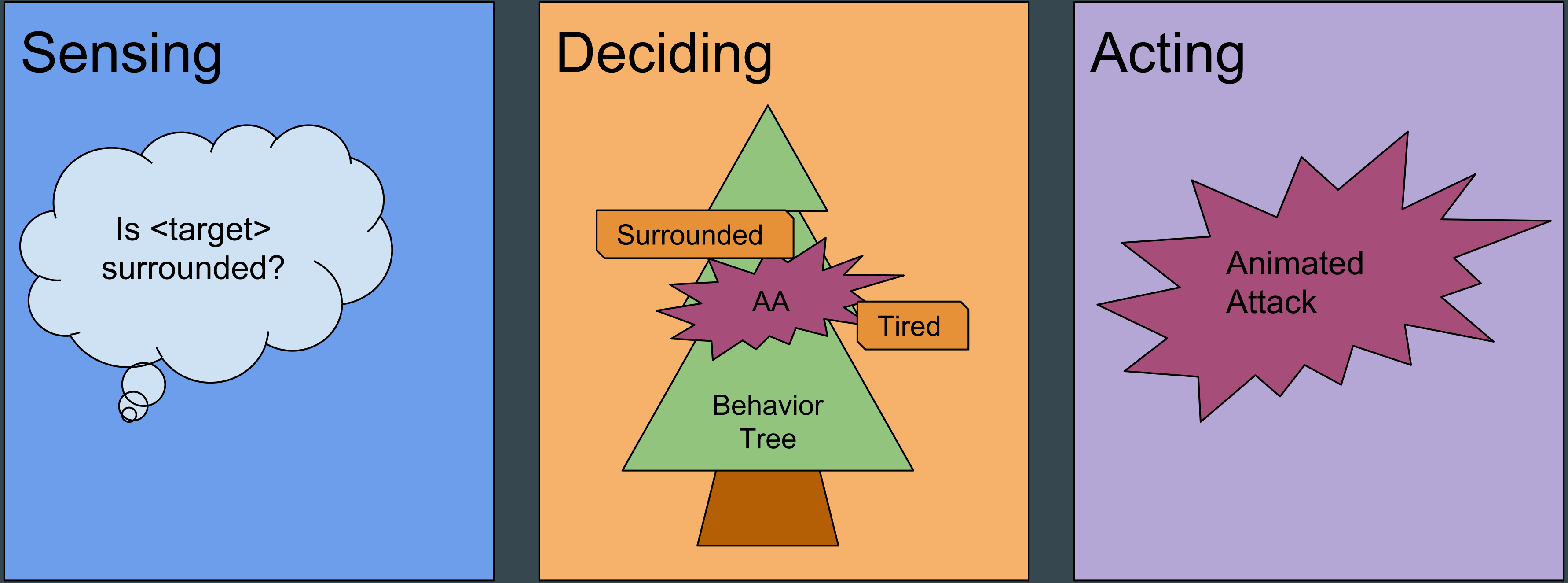


Forest Troll - Tree Trunk Tornado

- The details!
  - Should only attempt when surrounded
  - It’s a Point blank AOE attack
  - After completion, troll is “tired”
- So we make a new node for the behavior tree
  - OnStart, it checks enough enemies close by the troll
  - Do an animated attack, play an animation, sounds, triggers damage boxes
  - On completion, sets some data on the blackboard for “tired”
- Perfectly reasonable implementation
  - But we know what comes next

Ok, we failed. Now what?

Separation of Responsibility



Details, Guidelines and our solution

AI’s *World State* - the glue...

- Game State vs World State
- World State == Big Block O’ Data
- Guidelines
  - Every character should be able to have their own set
  - Keep this structure simple
- Evolve splits up this concept
  - The blackboard - enumeration of game state
  - The NEW whiteboard - rich data

Sensing

- Where data collection happens
- Guidelines
  - Modular and atomic
  - Characters should be able to have different sets of sensors.
  - Where the heavy lifting happens
  - Keep out of your deciding loop
- Evolve’s New Sensor Manager
  - Allows users to install sensors per character
  - Allows sensors to have a different stale time **per character type**
  - Manager burns through as many sensors it can in its allotted time

Acting

- Acting makes up the *operations* that the AI actually does
  - Your attacks, reloads, interactions, etc
- Guidelines
  - Operations should be agnostic to the reasons they are being done
  - Operations should be agnostic to the world state it references
  - Operations shouldn’t try to do more than one thing
  - Operations shouldn’t change your world state
- Evolve’s Acting
  - Our acting is done in our BT nodes

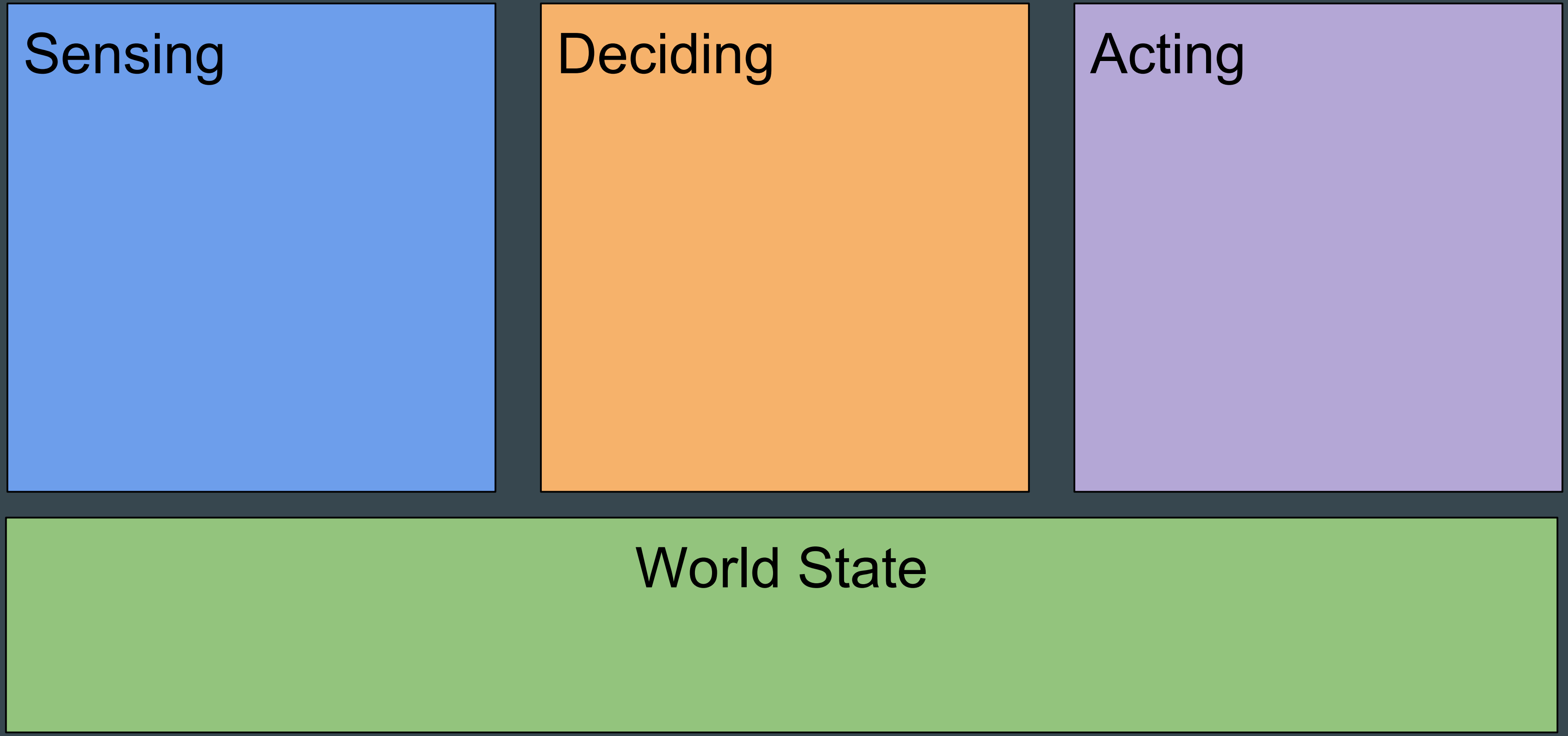
Deciding

- Where we make decisions
  - Behavior Systems (HTN, GOAP, BTs, FSM)
    - Kinda
- Guidelines
  - Deciding should be fast.
  - Should be able to annotate the acting components
  - Should be fast to edit.
- Evolve uses the BT described in Bill Merril's Game AI Pro article
  - Great article, go check it out

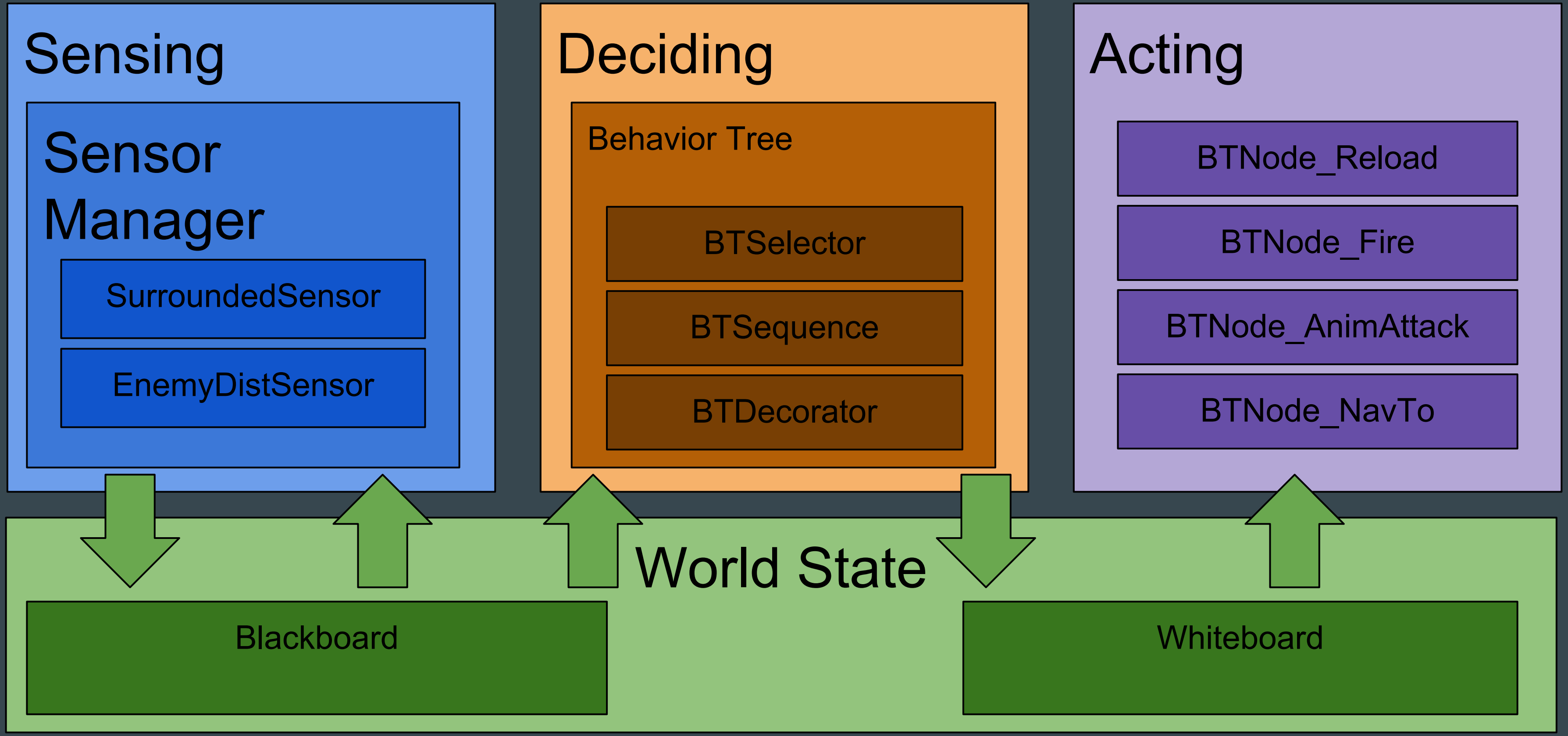
Blueprints - making it all stick together

- Evolve uses *blueprints* as a way to have one location where we can setup a character.
    - Load designer tunable data
    - Define the blackboard and install whiteboards
    - Install all the different components for our character
    - Connecting everything together
- This is how we built new AI without breaking legacy AI

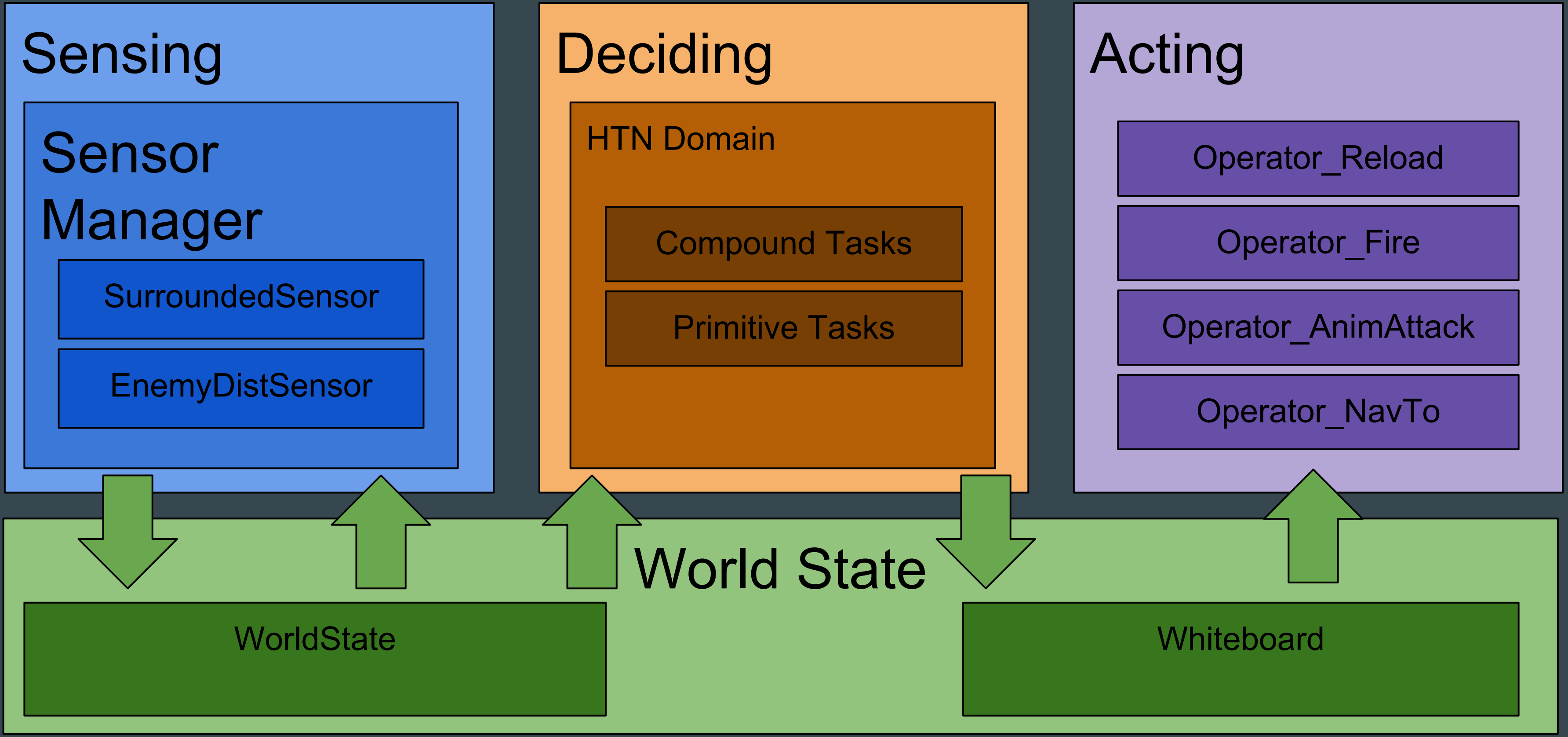
AI System Layout



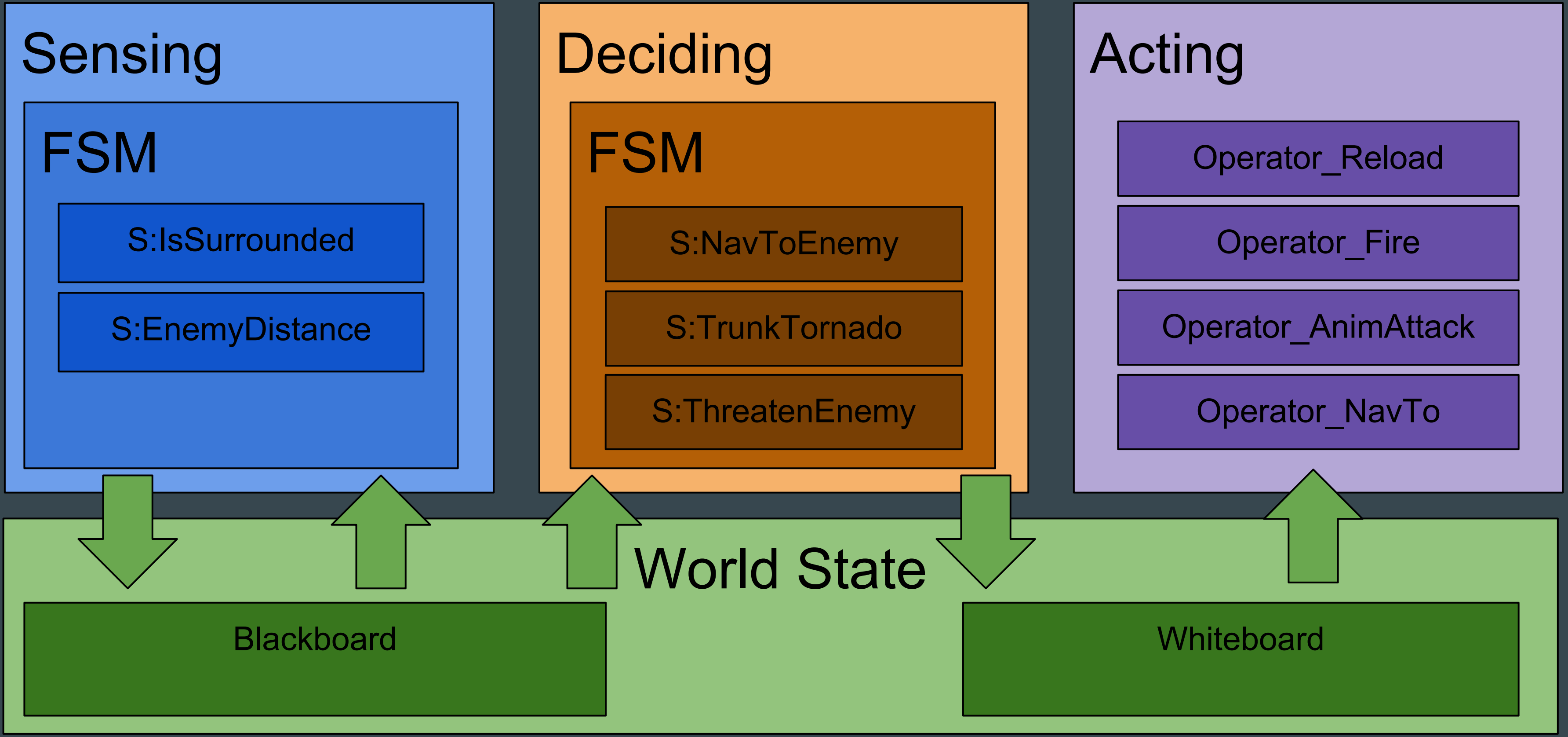
AI System Layout: Behavior Tree [Evolve]



AI System Layout: HTN Planner



AI System Layout: FSM



In Conclusion

- Modular system != Modular Characters
- Think in terms of responsibilities
- Make it easy for programmers to do the right work in the right place
- You can do it!

Thanks!

- Kevin and Chris for all their help!
- Justin Cherry for the awesome troll art!
- TRS AI team!
- Past AI teams

And my wife Liz, for listening to this talk over and over and over!