Scripting vs. Emergence

Martin Hafskjold Thoresen

What vs. What?

- Approaches to game design
- How is the game logic defined?
- Two approaches:
  - Scripting
  - Emergence

Scripting

- Predefined paths
- Relies on the designers ideas of what is fun
- Prone to inconsistencies
- Full creative control
- “Emulations” or “specific” system design

Emergence

«Fremvekst»

- Instead of hard-coding, make general rules
- Design types of objects and interactions
- Bullets break windows
- Consistency
- “Simulation”, or “systemic” system design
- The game emerges from the design
- Complex behaviour from simple rules

Developer Considerations

- Effort in Designing, Implementing and Testing
  - How do we program the behaviour we want?
- Effort in Modifying and extending
  - How hard is it to change some behaviour?
- Level of Creative Control
  - To what degree is it possible to control the system?
- Uncertainty and Quality Assurance
  - How hard is it to find and fix bugs?
- Ease of Feedback and Direction
  - How can we help the player in accomplishing their tasks?

Developer Considerations - Scripting

- Effort in Designing, Implementing and Testing
  - Must be done manually. Time consuming
- Effort in Modifying and extending
  - Explicit relationships with game elements for interactions
- Level of Creative Control
  - Full control.
- Uncertainty and Quality Assurance
  - No uncertainty or unexpected events. QA requires extensive testing
- Ease of Feedback and Direction
  - Easy to give feedback

Developer Considerations - Emergence

- Effort in Designing, Implementing and Testing
  - Bullet breaks window => projectiles break glass
  - Considerable initial effort
- Effort in Modifying and extending
  - Simple to extend, due to its general nature
- Level of Creative Control
  - Loss of control. Difficult to set up narratives
- Uncertainty and Quality Assurance
  - Uncertain, due to combinatorics. Requires extensive testing
- Ease of Feedback and Direction
  - A greater need. Hard

Player Considerations

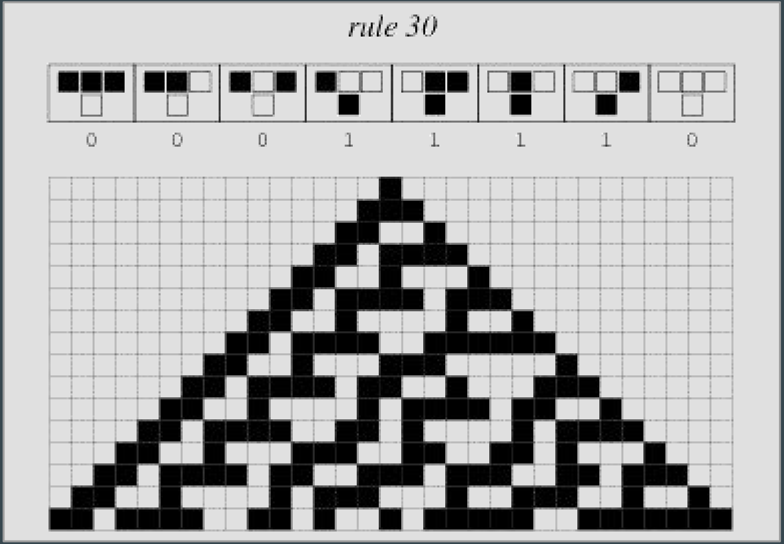
- Uphold suspension of disbelief
- Consistency and intuitiveness
  - Window breaking
  - Crate stacking
  - Exploding barrels
- Visually similar, functionally different
- Linearity, or low branching
- Replayability
- What the player wants to do --- what the designer wants to do

Techniques in Games - Scripting

- Finite State Machines (FSMs)
  - By far the most popular
  - Scales poorly
  - Difficult maintenance
- Scripting Languages
  - Simpler development
  - Artists and designers can script
  - Modding

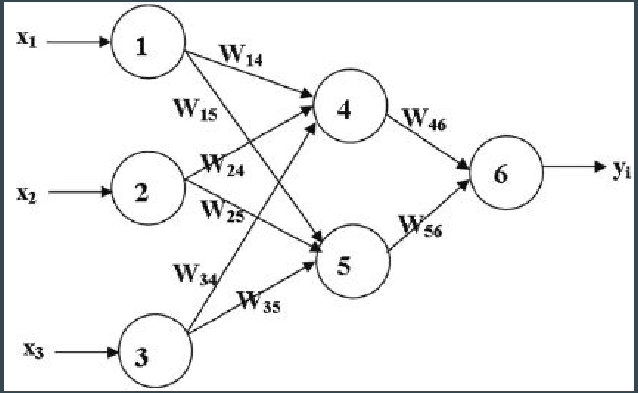
Techniques in Games - Emergence

- Flocking
  - Simulate group behavior
  - Boid
  - Separation: avoid crowding
  - Alignment: steer the boid toward average heading of local flockmates
  - Cohesion: steer the boid towards average position of local flockmates
- Cellular Automata (CA)
  - Grid of states, transition rule
  - Discrete time steps
  - Useful for fire, explosion, smoke, etc.



Techniques in Games - Emergence (2)

- Neural Networks (NN)
  - Brain inspired machine learning
  - Connected network of *units* and *weights*
  - Learns complex behaviour by training
  - Offline or online training?
- Evolutionary Algorithms (EA)
  - Evolution inspired
  - Merge and mutate
  - Parameter tuning: representation, population size, generations, fitness function, etc
  - Robust for large search spaces
  - Expensive



Where does this leave us?

- Scripting and Emergence are two extremes
- Both have benefits and drawbacks
- Is a sandbox/simulation even a game?
- Facilitate emergent interactions, script to set boundaries for story and game objectives