What is GlassBox?

• A new data-driven simulation engine for Maxis games
• Learn key lessons from the past
  • Power of data-driven simulation
  • Power of putting logic in game objects (The Sims)
• Being used to ship SimCity
Why?

• Bet for the future
• Adapt to a world dominated by the internet and non-PC devices
• Digital downloads and IAP vs. retail boxed product and one or two expansions
• DLG = Downloadable Gameplay
• Next Generation simulation
Why?

• Get to the gameplay more quickly
  • Build and deploy sim games more quickly
  • Easier iteration for higher quality
  • Allow significant post-ship updates

• Build ecosystems of simulation games
• Deploy same gameplay across multiple devices
Simulation Type

- Our past games have been primarily statistical
  - Heavily random-number based
  - Players good at rationalising random or even buggy behaviour as smart AI
- Good approach with limited CPU resources
  - But, makes it hard for player to understand what's going on with their sim (SimCity 4 traffic system)
  - Leads to gaps between visualisation and behaviour (Cars fading in and out in SimCity)
- We can do better
GlassBox Basics

- Resources + Units + Maps + Globals
- Combined with Rules
- In a Box

= $$$ Simulation!
Resources

• The basic currency of the game
  • Oil, coal, crops, wood, water...
  • Money, electricity, labour, pollution

• Resources come in **bins**:
  • Bin of resource $R$, has capacity $C$
  • Bin value is an integer, 0..$C$
  • Capacity is fixed
Units

• Represent *things*
  • houses, factories, even people

• A unit has state
  • A collection of resource bins

• Also a well-defined spatial extent
  • Bounding volume
  • Simulation footprint
Units

- **Food**: 20/30
- **People**: 4/10
- **Oil**: 40/100
- **Cars**: 2/4
- **Petrol**: 0/5
Maps

- Maps represent resources in the environment
  - Coal, oil, forest
  - But also air pollution, land value, desirability
  - Resources are *limited*
- Simple uniform size grids
  - Each cell is a resource bin
- Units interact with maps through their footprint
Maps
Maps

Resources
Units
Maps
Globals
Rules
Box
Globals

• Just a global set of resource bins
• Values associated with the game as a whole
• Next!
Rules

• We have the *nouns*, rules provide the *verbs*

• Rules operate on resources:
  • Move resources from one place to another
  • Convert resources to other resources
  • Have *inputs* and *outputs*

• Attached to the entity that runs them
Rule Example

- Money is converted to wood, if a person is available
  - Applied in its entirety
    - Only if the end result is valid
    - A rule can be applied multiple times

```plaintext
rule harvestWood
    Money in 10
    Wood out 2
    People in 1
    People out 1
end
```
Rule Example

- Money is converted to wood, if a person is available

- Applied in its entirety
  - Only if the end result is valid
  - A rule can be applied multiple times

```plaintext
rule harvestWood
Money in 10
Wood out 2
People in 1
People out 1
applyCount 1 10
end
```
Unit Rules

• Different targets:
  • **Local** (unit) bins
  • **Global** bins
  • **Map** cells covered by the unit
  • Bins in **nearby** units

• Can chain to other rules

• Trigger game actions
unitRule mustardFactory
  rate 10

  global Simoleans in 1
  local YellowMustard in 6
  local EmptyBottle in 1
  local BottleOfMustard out 1
end
Unit Rule Example

unitRule mustardFactory
  rate 10
  global Simoleans in 1
  local YellowMustard in 6
  local EmptyBottle in 1
  local BottleOfMustard out 1
end

- Run every 10 ticks
- Convert materials to product
Unit Rule Example

```
unitRule mustardFactory
  rate 10
  global Simoleans in 1
  local YellowMustard in 6
  local EmptyBottle in 1
  local BottleOfMustard out 1
end
```

- Run every 10 ticks
- Convert materials to product
Unit Rule Example

**unitRule** mustardFactory

* rate 10

* global Simoleans **in** 1

* local YellowMustard **in** 6

* local EmptyBottle **in** 1

* local BottleOfMustard **out** 1

* map Pollution **out** 5

end

- Run every 10 ticks
- Convert materials to product
- Emit some pollution
Unit Rule Example

unitRule mustardFactory
rate 10

global Simoleans in 1
local YellowMustard in 6
local EmptyBottle in 1
local BottleOfMustard out 1
map Pollution out 5
successEvent effect smokePuff
successEvent audio chugAndSlurp
end

- Run every 10 ticks
- Convert materials to product
- Emit some pollution
- Game feedback
Unit Rule Example

```
unitRule mustardFactory
  rate 10

  global Simoleans in 1
  local YellowMustard in 6
  local EmptyBottle in 1
  local BottleOfMustard out 1

  map Pollution out 5

  successEvent effect smokePuff
  successEvent audio chugAndSlurp

  onFail buyMoreMustard
end
```

- Run every 10 ticks
- Convert materials to product
- Emit some pollution
- Game feedback
- Chaining
Map Rules

• Operate on entire map, or a collection of random cells
• Run resource rule per cell
  • Can reference multiple maps at once
• Or, perform more specialised operations:
  • Diffusion (controlled by a second map)
  • Advection (e.g., by wind direction)
Map Rule Example

```plaintext
mapRule growGrass
   rate 200

map Soil atLeast 20
map Water in 10
map Nutrients in 1
map Grass out 5
end
```

- Grass will grow only where there’s soil, water, and nutrients
- Water and nutrients must be replenished
A Box

- Everything that makes up a game
- **Game Scripts**
  - Play Area and other properties
  - Unit types, Map types, Global bins
  - Rule scripts
- **Game State**
  - Bin and cell values
  - Unit locations

<table>
<thead>
<tr>
<th>Resources</th>
<th>Units</th>
<th>Maps</th>
<th>Globals</th>
<th>Rules</th>
<th>Box</th>
</tr>
</thead>
</table>

maxis
Key ideas

• Units contain their own simulation logic
  • You can drop in new units with new behaviours
  • Units can be combined to get aggregate behaviour

• Iteration, iteration, iteration
  • Hotloading for everything
  • Have an idea, implement, test, evaluate, ASAP

• Data-driven
  • Entirely defined by rule scripts and property lists
But wait... there's more

• This is enough to build a basic resource-based simulation game.
  • Have had fun building various mini games
  • Work in progress for future

• But not enough for SimCity-style sims
• Need: **Paths + Zones + Agents**
Paths
Paths

• **Points** connected by **Segments** make up **Paths** make up **Path Sets**

• Fully 3D, spline-based, rich set of operations

• Typically player created

• Curvy roads!

• But also: power lines, water pipes, flight paths
Zones

- Cover some well-defined area
- Run zone rules:
  - Create new units
  - Upgrade/downgrade existing units
  - Destroy units
- Provide "gardening" aspect of simulation
Zones
Zones
Zones
Zone Rule Example

zoneRule developHouses
timeTrigger Day 0.5

sample random -count 3

test global Builders greater 5
test map Forest is 0

createUnit -id Bungalows
end

• Try to create three houses a day
• Only if we have enough builders
• Only where the zone doesn’t overlap with forest
Agents

• Carry resources from one unit to another
  • Each has a set of resource bins
  • Do **not** run rules (10,000s of agents)

• Controlled by Transport Handlers
  • Agents handed over when emitted from unit
  • Handler responsible for delivering to a destination unit
Agents

• Created by unit rules
• Each agent is given a destination
  • Home, Work, Fire, Sickness
• Units can have **sinks** advertising the corresponding destinations
• Creation rule can set simple destination instructions
Unit Agent Rule

unitRule goToWork
options -sendTo Work -via Car -using Road

local People in 2
agent People out 2
end
Unit Agent Rule

unitRule goToWork
   options -sendTo Work -or Park
              -switchTo Home 10
              -repeatAfter 10
              -via Car -using Road

   local People in 2
   agent People out 2

end
Transport Handlers

• Predominantly path-oriented
  • Vehicles driving along paths
  • Resource flow through pipes

• But also
  • Helicopters, boats, aircraft
  • Free-routing sims
Path-based Routing

• Virtual Distance Field
  • D*-Lite based algorithm - wavefront updates
  • Calculates cost-to-nearest-sink at vertices
  • Steer towards vertex with least cost
  • No per-agent routing info

• Distance modified by
  • Sink strength: advertises a capacity
  • Modifiers such as congestion and speed limit
Virtual Distance Field
Virtual Distance Field

People 0/5
Virtual Distance Field

People 0/5

People 10/10
Virtual Distance Field

People 0/5

People 0/10
Virtual Distance Field

People 0/5

People 0/10

People 0/5
Virtual Distance Field
Virtual Distance Field

People 2/5

People 0/10

People 0/5
Virtual Distance Field

People 2/5

People 0/10

People 0/5
Virtual Distance Field

People 2/5

People 0/10

People 0/5
Virtual Distance Field

People 2/5

People 0/10

People 0/5
Virtual Distance Field

People 5/5
People 0/10
People 0/5
People 0/10
Virtual Distance Field

People 0/5

People 0/10

People 0/5
GlassBox Simulation

• Resources

• Units + Maps + Globals + Zones
  • Rules for each

• Paths + Agents
Online

• GlassBox built from ground up to support online
  • Data-driven means small downloads
  • Small upload bandwidth

• Game save is in the cloud: continuous save

• Play anywhere

• Rich online presence
Multiplayer

• Boxes communicate by sending **packages** back and forth
  • Online form of agents
• Can host boxes inside other boxes
  • SimCity regions are just another box
Online Buzzwords

• Asynchronous server model
  • No reliance on dedicated live server running to support your play session
  • Graceful degradation if we have server issues

• All-HTTP REST API

• Any cloud service supported: S3, EC2, etc.
Physics

• Assumed that units can move at will, and will be controlled by a physics simulation
• Simulator built around this assumption
• Avoid sim chugging to a halt during disasters
Visualisation

• Rather than visualise game statistics, show actual game state
  • Show cars instead of traffic density
  • Actual people in house rather than expected
• Ensure cause and effect is obvious:

What You See Is What You Sim
Conclusion

- The GlassBox simulation architecture is built out of very simple pieces
- But, the emergent behaviour is rich
- Now for SimCity...