<< RETURN

AI in Design: How AI Enables Designers

Brian Reynolds, President, Big Huge Games

AI in Design

- How AI enables Design (and vice versa)
- AI and Design should be created side-by-side
- Integrating Design and AI process
- Uses for AI: strategy, personality, content
- Case studies from games I've worked on
- AI techniques for Designers

Key Goals for Design

- Maximize interactivity
- Interesting & important choices
- Tension and asymmetry
- Replayability
- Compelling solo and multiplayer experience

Key Uses of AI

- Services (e.g. Path finding, physics)
- Strategy & tactics for computer *players*
  - Goals
  - Production
  - Armies
- Personality for computer *characters*
- Content generation, open-ended gameplay
  - Random maps, Random scenarios

Where Design and AI Overlap

- AI drives interactivity
  - Competitive, Cooperative, Diplomatic
- Add a feature == Must teach AI to play
  - Develop feature set that lends itself to good AI
- AI can provide asymmetrical landscape
- AI can generate content (replayability)
- Prototyping process

Prototyping AI *With* Design

- Improves the results you get from prototype
  - Allows you to play w/o multiplayer code

- Each independently benefits from process
  - Both need careful tuning!
  - "The AI is too dumb because…"
- Each draws inspiration from the other
  - AI behavior inspires new features
  - New features can provide basis for better AI

Tips for Prototyping AI

- Start *early* in development process.
- Just do it: start with something, anything!
  - Teach it to play one turn, then 10, then 100…
  - X=RND(3) constitutes valid prototype AI!
- Don't worry about creating "final" code yet
- Take a stab at AI for coolest design features
- Try to have AI confront player with choices

Prototyping Case Study: Diplomacy and Strategy

- (*Civilization* - 1991)
- *Colonization* - 1994
- *Civilization 2* - 1996
- *Alpha Centauri* - 1998
- *Rise of Nations* – 2003

Prototyping Case Study: Civilization and Colonization

- AI brings troops to your city even though you are "at peace"
- AI sits by while you bring troops near its cities
- Doesn't understand concept of territory
- Doesn't detect threats very well
- Not aware of your interactions with others

Prototyping Case Study: Civilization II

- Two-square "exclusion zone" around city
- Doesn't approach your cities while at peace
- Complains if you approach its cities
- Takes note of your interactions with others
- Still doesn't detect threats outside zone
- Helps itself freely to "your territory"

Prototyping Case Study: Alpha Centauri

- National borders introduced
  - Simple algorithm: which city is nearest
  - Visual representation of territory
- AI can tell whose "territory" unit is in
- AI can avoid hostile acts while at peace
- Can detect "adjacent" nations and threats
- Better diplomacy: "Hey, you're in my territory!"

Prototyping Case Study: Rise of Nations

- National Borders as centerpiece
- "Border pushing" as a strategy
  - Border algorithm to support "political game"
  - Strategic AI must understand how to do this
- Borders inspire additional AI & Design ideas
  - Attrition and Supply Wagons
  - Better army AI developed

AI and Design: Personality

- "Personality" for computer characters
  - Can enhance both story and replayability
- Works best with simple "polar" categories
  - Rush/Boom, Guns/Butter, Friendly/Enemy
  - Aim for simple, obvious, dramatic effects
- Requires careful tuning: start early!

Personality Case Study: Alpha Centauri

- Social Engineering
  - Player makes social policy
  - Policy produces effects in game world
- Diplomacy
  - Faction leaders have established social agendas
  - Player policies affect relationships with leaders
- Result
  - Diplomacy isn't just about strong/weak
  - Faction leaders seem more multi-dimensional

Content Generation:

Using AI to Do Design

- Random Maps
  - Asymmetry vs. Balance
- Random Scenarios & Campaigns
  - Greatly enhance replayability
  - Create scenario based on effects of prior actions
  - Experience true “Fog of War”
  - Example: Conquer the World

Content Generation: Key Tasks

- Fill up the world with fun toys!
- Create asymmetry and tension
  - Variety from encounter to encounter
  - Force players to commit themselves
- Create balanced playing field
- Integrate seed data with “random” factors

AI and Design: The “Don’ts”

- Wait until “design doc” is complete to start AI
- Create AI (or Design) in a vacuum
- Be afraid to get started on AI
- Get bogged down in complexity
- Spend too much time up front on algorithms

AI and Design: The “Do’s”

- Start early and prototype
- Form close relationship between AI & Design
- Prototype AI and Design simultaneously
- Aim for crisp, simple game effects
- Teach AI to use newly designed features
- Use AI to generate open-ended content
- Use AI to drive interactivity

Content Generation: Basic Tools

- Random Numbers
  - The Designer’s “basic tool”
- Fractals
  - “Clumpy” random numbers
  - Emphasize asymmetry over balance
  - Work best over simple, linear domains
- CRC
  - Randomize list-traversal without duplication

– Balanced resource spread w/o obvious pattern

Code Sample: “CRC Random”

```
//
// CRC “Random” – Brian Reynolds
//
// Useful for semi-random (scattered) traversal of a list,
// without duplication of any numbers. Useful for spreading
// resources on random maps, or simply scrambling city name
// list, etc.
//
// crc_poly -> CRC “polynomial” (each bit represents that
//             power of “x”). Polynomials provided for 8,
//             12, 16 bits.
//
// crc_seed -> Where to start in your list. If your list
//             size is smaller than the available bit-width,
//             make sure your initial seed is within the
//             valid range of entries. Also the send can’t
//             be zero since the range is 1-n, not 0-n.
//
// crc      -> The current active index/value at any given
//             point in the loop. Note this will never equal
//             0, so an 8-bit CRC returns values 1-255.
//             Initialized to equal “crc_seed”, and when
//             it again equals crc_seed, it means one
//             complete loop through the available range
//             has been performed.
//
crc_poly = 0xb8; // 8-bit (12-bit:0xca0, 16-bit:0xb400)
crc_seed = ((WORLD.seed & 255) % num_rivers) + 1;
crc      = crc_seed; // Must be non-zero
do {
  do {
    if (crc & 1) {
      crc  = crc >> 1;
      crc ^= crc_poly;
    } else {
      crc  = crc >> 1;
    }
    if (crc == crc_seed) break; // Stop after full cycle
  } while (crc > num_rivers);

  // Perform functions here...
  bed_x = river_beds[crc - 1].x;
  bed_y = river_beds[crc - 1].y;
  ...
} while (crc != crc_seed);
```

AI in Design

- How AI enables Design (and vice versa)
- AI and Design should be created side-by-side
- Integrating Design and AI process
- Uses for AI: strategy, personality, content
- Case studies from games I've worked on
- AI techniques for Designers

Key Goals for Design

- Maximize interactivity
- Interesting & important choices
- Tension and asymmetry
- Replayability
- Compelling solo and multiplayer experience

Key Uses of AI

- Services (e.g. Path finding, physics)
- Strategy & tactics for computer *players*
  - Goals
  - Production
  - Armies

```
if (TECHTYPE.where == UNIVERSITY) {
  // these are very very important
  val *= 1000;
}
```

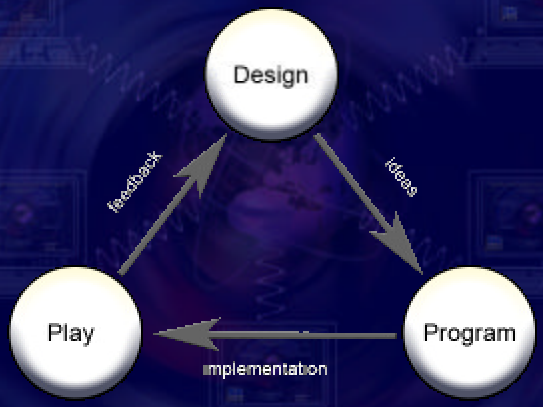
- Personality for computer *characters*
- Content generation, open-ended gameplay
  - Random maps, Random scenarios

Where Design and AI Overlap

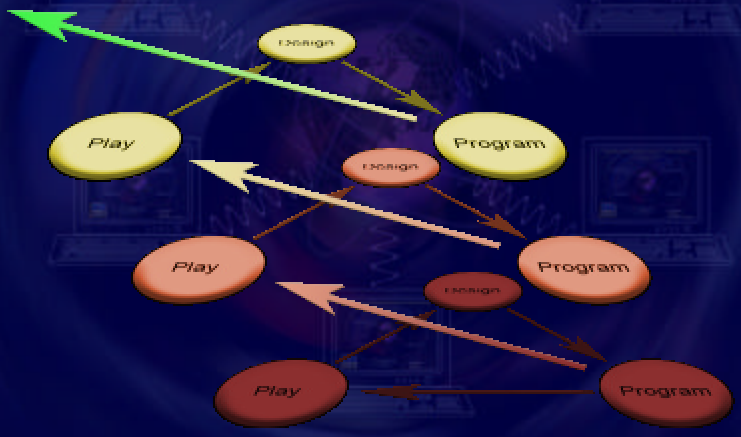
- AI drives interactivity
  - Competitive, Cooperative, Diplomatic
- Add a feature == Must teach AI to play
  - Develop feature set that lends itself to good AI
- AI can provide asymmetrical landscape
- AI can generate content (replayability)
- Prototyping process

```
// Dixitque Deus: Fiat Lux!
map.make(NUM_NATIONS, map.seed);
```

Creating Gameplay



Prototyping Process



Prototyping AI *With* Design

- Improves the results you get from prototype
  - Allows you to play w/o multiplayer code
- Each independently benefits from process
  - Both need careful tuning!
  - "The AI is too dumb because…"
- Each draws inspiration from the other
  - AI behavior inspires new features
  - New features can provide basis for better AI

Tips for Prototyping AI

- Start *early* in development process.
- Just do it: start with something, anything!
  - Teach it to play one turn, then 10, then 100…
  - X=RND(3) constitutes valid prototype AI!
- Don't worry about creating "final" code yet
- Take a stab at AI for coolest design features
- Try to have AI confront player with choices

```
//Decide which way to turn, but be sure not to change your mind
orders->sharp_turn = rnd(2);
```

Prototyping Case Study: Diplomacy and Strategy

- *(Civilization* - 1991)
- *Colonization* - 1994
- *Civilization 2* - 1996
- *Alpha Centauri* - 1998
- *Rise of Nations* - 2003

```
// try not to build in your ally's territory, it's rude
if (world.get_who(wx, wy) != who) val /= 3;
```

Prototyping Case Study: Civilization and Colonization

- AI brings troops to your city even though you are "at peace"
- AI sits by while you bring troops near its cities
- Doesn't understand concept of territory
- Doesn't detect threats very well
- Not aware of your interactions with others

```
// In the following situations we should
// definitely avoid aggression:
if (weak[whom] || (LEADER2.attack > attack*3)) {
  if (!courageous) aggression[whom] = false;
}
```

Prototyping Case Study: Civilization II

- Two-square "exclusion zone" around city
- Doesn't approach your cities while at peace
- Complains if you approach its cities
- Takes note of your interactions with others
- Still doesn't detect threats outside zone
- Helps itself freely to "your territory"

```
// Decide whether to demand tribute here (or just declare war)
if ((LEADER2.betrayals < 4) && !rush) {
  action_demand_tribute (whom, WEALTH, min (wealth, BUCKETS));
}
```

Prototyping Case Study: Alpha Centauri

- National borders introduced
  - Simple algorithm: which city is nearest
  - Visual representation of territory
- AI can tell whose "territory" unit is in
- AI can avoid hostile acts while at peace
- Can detect "adjacent" nations and threats
- Better diplomacy: "Hey, you're in my territory!"

```
// tell the city it is near a border
CITY2.bordering |= (1 << who);
CITY2.bordering |= (1 << who2);
```

Prototyping Case Study: Rise of Nations

- National Borders as centerpiece
- "Border pushing" as a strategy
  - Border algorithm to support "political game"
  - Strategic AI must understand how to do this
- Borders inspire additional AI & Design ideas
  - Attrition and Supply Wagons
  - Better army AI developed

```
// Stop pushing cities in a straight line, which AI was tending to do
if (LEADER.city_num >= 3) temp = min (temp, 10);
```

AI and Design: Personality

- "Personality" for computer characters
  - Can enhance both story and replayability
- Works best with simple "polar" categories
  - Rush/Boom, Guns/Butter, Friendly/Enemy
  - Aim for simple, obvious, dramatic effects
- Requires careful tuning: start early!

```
// Border skirmish or Capital attack?
if (PERSONALITY.raid < 0) {
  if (CITY2.is_capital()) val *= 2;
} else if (PERSONALTY.raid > 0) {
  if (CITY2.near_border(who)) val *= 2;
}
```

Personality Case Study: Alpha Centauri

- Social Engineering
  - Player makes social policy
  - Policy produces effects in game world
- Diplomacy
  - Faction leaders have established social agendas
  - Player policies affect relationships with leaders
- Result
  - Diplomacy isn't just about strong/weak
  - Faction leaders seem more multi-dimensional

Content Generation: Using AI to Do Design

- Random Maps
  - Asymmetry vs. Balance
- Random Scenarios & Campaigns
  - Greatly enhance replayability
  - Create scenario based on effects of prior actions
  - Experience true "Fog of War"
  - Example: Conquer the World

```
// Check if either side is bringing allies to this scenario
ally  = CONQUEST_NODE2.num_allied_armies(node, allies);
ally2 = CONQUEST_NODE2.num_allied_armies(node2, allies2);
```

Content Generation: Key Tasks

- Fill up the world with fun toys!
- Create asymmetry and tension
  - Variety from encounter to encounter
  - Force players to commit themselves
- Create balanced playing field
- Integrate seed data with "random" factors

AI and Design: The "Don'ts"

- Wait until "design doc" is complete to start AI
- Create AI (or Design) in a vacuum
- Be afraid to get started on AI
- Get bogged down in complexity
- Spend too much time up front on algorithms

AI and Design: The "Do's"

- Start early and prototype
- Form close relationship between AI & Design
- Prototype AI and Design simultaneously
- Aim for crisp, simple game effects
- Teach AI to use newly designed features
- Use AI to generate open-ended content
- Use AI to drive interactivity

Questions & Answers

```
// Border skirmish or Capital attack?
if (PERSONALITY.raid < 0) {
  if (CITY2.is_capital()) v
} else if (PERSONALTY.raid
  if (CITY2.near_border(
}
```

```
if (TECHTYPE.where == UNIVERSITY) {
  // these are very very important
  val *= 1000;
```

```
// Check if either
ally  = CONQUEST_NO
allies to this scenario
ies(node, allies);
ies(node2, allies2);
```

```
// Don't go crazy on boats
if (reg_combat[reg2] > (en
if (reg_combat[reg2] > (en
sn't have many
inue;
ue;
```

```
// Decide if this a
to release into the world
int Army::release_mustering (void);
```

```
if (diplo == WAR) {
  msg = S_ARRAY("$DUBYA0 and $SADDAM1 have gone to war.", 3111);
} else if (diplo == PEACE) {
```

AI and Design: How AI Enables Designers

2004 Game Developers Conference

Presented by
Brian Reynolds, President
Big Huge Games

BIG HUGE GAMES

Content Generation: Basic Tools

- Random Numbers
  - The Designer's "basic tool"
- Fractals
  - "Clumpy" random numbers
  - Emphasize asymmetry over balance
  - Work best over simple, linear domains
- CRC
  - Randomize list-traversal without duplication
  - Balanced resource spread w/o obvious pattern

Code Sample: "CRC Random"

```
crc_poly = 0xb8; // 8-bit (12-bit:0xca0, 16-bit:0xb400)
crc_seed = ((WORLD.seed & 255) % num_rivers) + 1;
crc      = crc_seed; // Must be non-zero
do {
  do {
    if (crc & 1) {
      crc  = crc >> 1;
      crc ^= crc_poly;
    } else {
      crc  = crc >> 1;
    }
    if (crc == crc_seed) break; // Stop after full cycle
  } while (crc > num_rivers);

  // Perform functions here...
  bed_x = river_beds[crc - 1].x;
  bed_y = river_beds[crc - 1].y;
  ...
} while (crc != crc_seed);
```