Comp 388/488 - Game Design and Development

Spring Semester 2018 - Slides - week 7

Dr Nick Hayward

DEV Week Assessment

- demo and project report
  - *due on Friday 16th March 2018 @ 2.45pm*
- anonymous peer review
  - *similar to user comments and feedback*
  - *chance to respond to feedback before final project*

DEV week overview...

- brief presentation or demonstration of current project work
- 30% of final grade
- ~ 10 minutes per group
- analysis of work conducted so far
  - *eg: during semester & DEV week*
- presentation, demonstration, or video overview...
- outline current state of game
- show prototypes and designs
- explain what works & does not work
- outline research conducted
- anything else considered relevant to your research or development...
- show any mockups, prototypes, patterns, and designs

Games and Dynamics

systems and evolution - intro

- a game's players, their type, number, reactions, behaviours &c.
  - *may also be a reflection of the game system itself*
- systems may often display complex and unpredictable results
  - *e.g. when set in motion as part of the broader, general gameplay*
- such systems are not inherently predicated on complexity and scope
- good examples of simple rule sets and patterns
  - *produce unpredictable results as they are set in motion*
- we may see such patterns on a regular basis, e.g. in the natural world
  - *complex systems emerging due to collaboratrive structures, e.g.*
  - *ant colony*
  - *bee hive and pollen collection*
  - *swarm intelligence of a confusion of wildebeest*
- human consciousness may be the product of such systems
  - *commonly referred to as emergent systems*
- well-known experiment in emergence was conducted by John Conway in the 1960s
  - *particularly useful and interesting for us as game designers and developers*
- Conway was particularly curious about the working systems of rudimentary elements
  - *how did such elements work together based upon a set of defined, simple rules...*
- he wanted to clearly demonstrate this phenomenon at its simplest level
  - *e.g. in a defined 2D space, such as a known chess board*
- he tested various ideas and concepts
  - *considered ideas such as on and off logic for squares/cells on a board*
  - *logic based on rules for adjacent squares/cells*
- he continued his experiments and tests
  - *in a similar manner to a game designer*
  - *toyed with various sets of rules for several years*

Games and Dynamics

systems and evolution - simple rules

rules

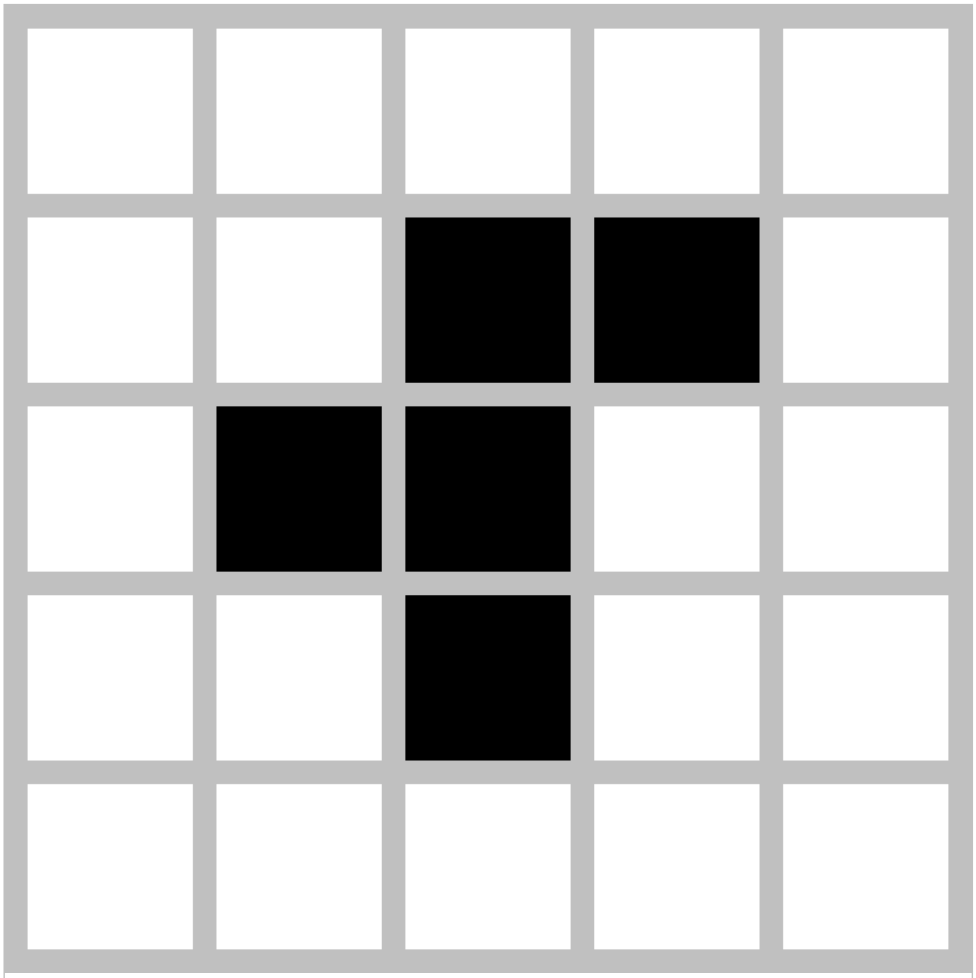
- Birth
  - *if a cell is unpopulated, and surrounded by exactly 3 populated cells, this cell will be populated in the next generation*
- Death by loneliness
  - *if a cell is populated, and surrounded by fewer than 2 populated cells, this cell will be unpopulated in the next generation*
- Death by overpopulation
  - *if a cell is populated, and surrounded by at least 4 populated cells, this cell will be unpopulated in the next generation*
- emergent system finally converged on the above set of rules
- Conway, and some of his colleagues at Cambridge, began populating their chess board with pieces
  - *then tested their rules by hand*
- started to learn about this system
  - *and the very nature of emergent, almost evolving systems*
- quickly realised that different starting conditions had a noticeable impact on a system's evolution
- realised that the complexity of such start conditions might have a side-effect on the patterns created
  - *many simply failing to survive and evolve*
- a particularly interesting discovery became known as the **R Pentomino** configuration
  - *Richard Guy, an associate of Conway, became fascinated with this particular configuration*
- Guy tested their defined rules for more than a hundred generations
  - *started to observe various patterns emerging*
- a regular mix of shapes and patterns emerging
  - *Guy noticed that his shapes appeared to moving, effectively walking across the board*
  - *he exclaimed at this discovery,*

> *Look, my bit's walking*

- Poundstone, W. *Prisoner's Dilemma.* Touchstone. New York. 2002.
- Guy continued to test and work on this configuration
  - *until he was able to get this pattern to actually walk across the room*
  - *and then out the door...*
- Guy's discovery became known simply as a **glider**

Image - Systems and Evolution

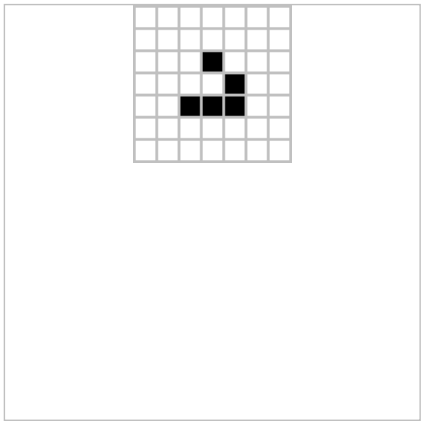
R-Pentomino



R Pentomino

Image - Systems and Evolution

glider evolution



- interactive demo - glider

Video - Nature and Systems

BBC - The Code.

Source - BBC - The Code, YouTube

Games and Dynamics

systems and evolution - examples

- such simple emergent systems demonstrated
  - *benefits and application of simplicity in rules and patterns*
  - *their ability to evolve into life-form style patterns*
- such systems had the potential to evolve and develop with each generation
  - *particularly interesting and useful to us as game designers*
- may start to add such techniques to help make our games
  - *more realistic, evolving, and unpredictable for our players...*
- example games using these techniques include:
  - *Black and White*
  - *Grand Theft Auto (v3 onwards)*
  - *Halo*
  - *Oddworld: Munch's Oddysee*
  - *The Sims*
  - ...

Video - Emergent Behaviour in Game AI

GDC - Emergent Behaviour in Game AI

Source - GDC, YouTube

quick exercise

A quick exercise to consider evolution in systems,

- *Traveling Salesman Problem*
- evolution of simple systems
- swarm/hive intelligence
- interaction in systems

requirements

- consider the above, and how they might interact in a system to evolve an optimal solution to a problem
- consider application of such simple systems and evolution in a game environment

Video - Algorithms and Evolution

BBC - Algorithms and evolution.

Source - Algorithms, YouTube

Games and formal elements

importance of objectives

- objectives may help establish different requirements and goals in a game
  - *helping a user to achieve results within the confines of the rules of the game*
- objectives may seem challenging and difficult
  - *need to be correctly designed relative to a game's rules*
  - *they should also seem achievable to a player*
- a game's objective may also help set the tone for gameplay and interaction
  - *e.g. objective of most platform games different from sports-based game*
  - *tone for each of these games becomes a reflection of the objective*
- use of objectives in games is not limited to just the overall game itself
- may consider defining an objective for different player roles
  - *or perhaps as mini challenges within our games*
- each level may define its own objective
  - *such as completing a level as fast as possible*
  - *collecting all available options (coins, for example) on another*
- choice of such objectives needs to be considered carefully
- each will affect not only the formal system of our game
  - *but also the dramatic aspect*
- good integration of objectives in the premise or story of a game
  - *helps strengthen dramatic aspects*
  - *e.g. Legend of Zelda*

Games and formal elements

a consideration of procedures

- we may start to see a few common actions that exist across multiple genres
- these often include the following:
  - *an action to start the game*
    - specific procedure required to initiate gameplay...
  - *ongoing actions and procedures*
    - e.g. common, persistent actions that continue, repeat &c. as part of the game
  - *reserved or special actions*
    - e.g. actions that may be required and executed due to a given condition or game requirement
  - *actions to conclude or resolve*
    - e.g. resolve actions at certain points within the game, or at the end of the game itself...
- for video games, incl. consoles, mobile, PC...
  - *such actions and procedures closely associated player interactions*
  - *e.g. given key combinations or controller buttons*
  - *perhaps tapping particular options on the screen itself*
  - *or moving a mobile device to control certain actions...*
- consider Super Mario Bros.
  - *we may clearly identify controls for given actions and procedures*
    - expected usage for directional buttons
    - option to jump or swim with the **A** button, &c.

Games and formal elements

procedures in development

- procedures also play a key role in the way we develop our games
- we can add procedures within the logic of our game
  - *to monitor certain ongoing states, user interaction, updates, rendering...*
- these procedures are working in the core of our game
  - *responding to changes in state*
- e.g. a player completes a puzzle within the main game
  - *need to monitor the ongoing puzzles responses*
  - *check the player input and interactions*
  - *then update the state of the game in response to a success or failure result*
- we're effectively checking whether a given action succeeds or not
  - *then, determine impact this may have on the game itself*
- such procedures and actions are naturally limited by real-world constraints
  - *e.g. performance of the underlying system, controllers, interaction options, screen...*
- may need to tailor such requirements to match the type of game we're developing
  - *and the target audience...*

Games and formal elements

rules and game concepts

- as we define and formalise rules for our games
  - *need to consider more than simply the gameplay itself*
- objects in games, and concepts embedded in gameplay structures
  - *require defined limitations and rules*
- game objects
  - *characters, weapons, vehicles, obstacles...*
  - *may be derived or inspired by real world objects*
- objects may come with the perception of existing limitations and rules
  - *a player knows what these objects can and cannot do in the real world*
- we may use these real world objects as inspiration
  - *starting points for our game's objects*
  - *not inherently limited or defined by them*
  - *may modify as befits the requirements of our game, and its gameplay*
- game context will be a determining factor in development of our objects
- objects may also be developed as a group of properties and variables
  - *together form the whole from varying requirements*
- in a world of chivalry, knights, ogres, and other fantastical creatures
  - *we may still create concepts and objects that unify these characters*
  - *from base objects, we can simply inherit and modify as needed*
- e.g. we may require various characters to ride
  - *on horseback, or perhaps astride an elephant, or even a fictional dragon &c.*
- our objects may be abstracted to include known attributes
  - *which can then be used as the parent*
  - *use for multiple real and imagined objects, characters within our game*

Games and formal elements

rules, objects, and updates...

- as developers and designers
  - *need to ensure a balance between maintaining game objects and variables*
  - *creating an intuitive update for our users*
- unlikely our player will want to keep a manual tally of such updates
  - *need to consider how we may allow them to quickly and easily intuit game objects*
- for example, we may need to
  - *maintain a running total of game objects, such as coins, lives, energy levels*
  - *correctly inform the player of any updates*
- a player should be able to quickly learn the nature of these objects
- if they're too difficult or complex
  - *need to consider how this affects our player's gaming experience*
- also need to ensure that there is sufficient isolation between different objects
- a player should be able to discern differences without too much effort or guesswork
- updates may also be influenced by known restrictions in the game's rules
  - *useful in many respects*
  - *e.g. relative to boundaries, objectives, and objects themselves*
- by establishing rules, e.g.
  - *to restrict objects and their attributes*
- rules help create a known scale for state within our game
- player has defined restrictions
  - *they know what they can and can't do*
  - *risk and reward is set in the game's logic and gameplay*

Python and Pygame - Game Example 1

render text to a game window - intro

- drawing text to a game window in Pygame can become a repetitive process
  - *in particular, as part of each window update*
- we may abstract this underlying game requirement to a text output function

```
 # text output and render function - draw to game window
def textRender(surface, text, size, x, y):
    # specify font for text render
    ...
```

- start by specifying a surface where we need to draw the text
  - *plus text to render, its size, and coordinates relative to surface*
- need to specify a font for the text to be rendered
  - *reliant upon installed fonts for user's local system*
- use a font-match function with Pygame
  - *helps abstract specification of exact font to a relative name*

```
# specify font name to find
font_match = pygame.font.match_font('arial')
```

- Pygame will search local system for a font with the specified name
- use specified font to create an object for the font
  - *we need this object to render text in the game window*

```
# specify font for text render - uses found font and size of text
font = pygame.font.Font(font_match, size)
```

Python and Pygame - Game Example 1

render text to a game window - text drawing

- text we'll be adding to the game window needs to be drawn
  - *drawn effectively pixel by pixel*
- Pygame calculates drawing for each pixel
  - *creates the specified text in the required font*
- start by specifying a surface to draw the required pixels for the text, e.g.

```
# surface for text pixels - TRUE = anti-aliased
text_surface = font.render(text, True, WHITE)
```

- we're specifying where to draw the text
  - *the text to draw to the game window*
  - *a boolean value for anti-aliasing of text*
  - *and the text colour*
- need to calculate a rectangle for placing the text surface, e.g.

```
# get rect for text surface rendering
text_rect = text_surface.get_rect()
```

- then specify where to position our text surface
  - *relative to defined `x` and `y` coordinates, e.g.*

```
# specify a relative location for text
text_rect.midtop = (x, y)
```

- text is then added to the surface using the standard `blit` function, e.g.

```
# add text surface to location of text rect
surface.blit(text_surface, text_rect)
```

render text to a game window - text draw function

- overall text draw function is now as follows,

```
# text output and render function - draw to game window
def textRender(surface, text, size, x, y):
    # specify font for text render - uses found font and size of text
    font = pygame.font.Font(font_match, size)
    # surface for text pixels - TRUE = anti-aliased
    text_surface = font.render(text, True, WHITE)
    # get rect for text surface rendering
    text_rect = text_surface.get_rect()
    # specify a relative location for text
    text_rect.midtop = (x, y)
    # add text surface to location of text rect
    surface.blit(text_surface, text_rect)
```

- call this function whenever we need to render text to our game window
- in `draw` section of our game loop, now add the following call, e.g.

```
# draw text to game window - game score
textRender(window, str(game_score), 16, winWidth / 2, 10)
```

Python and Pygame - Game Example 1

render text to a game window - add a game score

- common example of rendering text in a game window
  - *simply output a running score for the player*
- start by adding an initial variable to record the player's score, e.g.

```
# initialise game score - default to 0
game_score = 0
```

- then allow a player to score points for each projectile collision on a mob object
  - *e.g. laser beam hit on an asteroid*
  - *fun to set variant points relative to size of mob object*
- if we use the radius of each mob object
  - *perform a quick calculation for each* `collision`
  - *work out points per asteroid, e.g.*

```
# calculate points relative to size of mob object
game_score += 40 - collision.radius
```

- relative to the recorded `collision`
  - *simply get the radius per hit mob object*
  - *then minus from a known starting value*

resources

- notes = drawing-text.pdf
- code
- drawingtext1.py (game example with score)
- drawingtext2.py (abstracted - simple rendered text)

game example

- shooter0.8.py
- draw text to the game window
- keep a running score for collisions with a projectile
  - *player shoots and destroys an asteroid*
  - *score is calculated relative to size of mob object - radius...*
- score is rendered to top of game window
  - *update for each successful hit*

Video - Shooter 0.8

render text for a game score

Game designers

Designer example - Will Wright

- Wright is a veteran American game designer
  - *best known for his work on The Sims*
- *The Sims* was originally released in 2000
  - *led to countless versions, spin-offs &c.*
  - *driven a genre more interested in participation than a definitive win*
- as a co-founder of *Maxis*, and then later part of EA
  - *Wright also developed the game Spore*
- he's often referred to as a designer of *software toys* instead of traditional games
  - *a consideration of the non-traditional structure employed for many of his games*
- he's also been a passionate developer of, and advocate for, emergent and adaptive systems
- Wright has continued to develop this concept for many of his games
  - *his legacy is evident in games such as Spore, The Sims 3 and The Sims 4*
- Wright has tried to use these systems with their simple rules and definitions
  - *to provide the possibility for the development of complex, detailed outcomes*

Resources

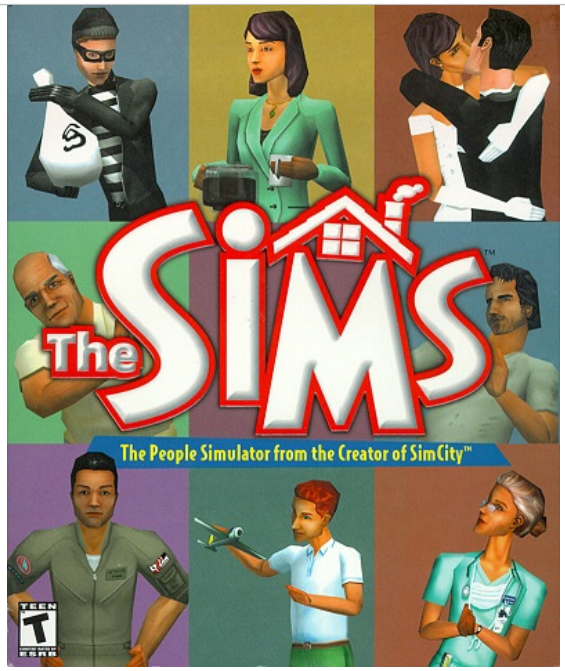
- Maxis
- The Sims
- Spore
- Will Wright

Image - Will Wright

Will Wright

Image - Will Wright

The Sims and Spore



The Sims

Demos

- pygame drawing text
  - *drawingtext1.py (game with text)*
  - *drawingtext2.py (simple rendered text)*
- pygame - Game 1 Example
  - *shooter0.8.py*

Games

- Maxis
- The Sims
- Spore

Game notes

- Pygame
  - *drawing-text.pdf*

References

- Conway and Life Patterns
  - *LifeWiki*
  - *Richard Guy*
  - *Glider*
  - *interactive demo - glider*
- Huizinga, J. *Homo Ludens: A Study of the Play-Element in Culture*. Angelico Press. 2016.
- Poundstone, W. *Prisoner's Dilemma.* Touchstone. New York. 2002.
- Will Wright

Videos

- BBC - Algorithms - YouTube
- BBC - The Code, YouTube
- GDC - Emergent Behaviour in AI - YouTube