Comp 388/488 - Game Design and Development

Spring Semester 2018 - Slides - week 12

Dr Nick Hayward

health and status - intro

- may add a status bar for a player's health, lives, &c.
- then dynamically update it relative to a defined health value
  - *e.g. a percentage value we decrement per collision*
- current game only gives a player one chance to shoot and destroy mob objects
  - *in effect, player currently has one life*
  - *one player life is not expected for most shooter style game...*
- may now consider monitoring and updating the status of a player's health
  - *e.g. as they are hit by advancing mob objects*
- protect our player, and their ship, using a Star Trek style shield
  - *may offer full protection initially*
  - *then incrementally weaken with each hit from a mob object*
  - *weakens until it eventually fails at value 0*
- set a default for this shield in the `Player` class,

```
# set default health for our player - start at max 100% and then decrease...
self.stShield = 100
```

health and status - collisions and shields

- need to modify our logic for a mob collision to ensure we handle such objects better
  - *may now reflect a decrease in the player's shield, health...*
- instead of allowing a mob object to continue after it has collided with the player
  - *now need to remove it from the game window*
- if we don't update this boolean to `True`
  - *each mob object will simply continue to hit the player*
  - *hit registered as it moves, pixel by pixel, through the player's ship*
  - *single hit would quickly become compounded in the gameplay*
- update for this check, e.g.

```
# add check for collision - enemy and player sprites (True = hit object is now deleted from game window)
collisions = pygame.sprite.spritecollide(player, mob_sprites, True, pygame.sprite.collide_circle)
```

- as our player may be hit by multiple mob objects
  - *also need to update our check from a simple conditional to a loop*
  - *check possible collisions...*

```
# check collisions with player's ship - decrease shield for each hit
for collision in collisions:
    # decrease player's shield for each collision
    player.stShield -= 20
    # check overall shield value - quit game if no shield
    if player.stShield <= 0:
        running = False
```

health and status - replace mob objects

- we still have an issue with losing mob objects
  - *if they collide with the player's ship*
  - *follows same underlying pattern as player's laser beam firing on mob objects*
- need to create a new object if it is removed after a collision
- a familiar pattern we may now abstract
  - *creation of mob objects to avoid repetition of code, e.g.*

```
# create a mob object
def createMob():
    mob = Mob()
    # add to game_sprites group to get object updated
    game_sprites.add(mob)
    # add to mob_sprites group - use for collision detection &c.
    mob_sprites.add(mob)
```

- simple abstracted function allows us to easily recreate our mob objects
  - *by creating a mob object*
  - *adding it to the overall group of* `game_sprites`
  - *then the specific group for the game's* `mob_sprites`
- then call this function if a mob object collides with a projectile, player's ship...
- also call this function when we initially create our new mob objects

```
# create a new mob object
createMob()
```

Python and Pygame - Game Example 1

health and status - health status bar

- already defined a default maximum for our player's shield
  - *now start to output its value to the game window*
- we could simply output a numerical value
  - *as we did for the player's score*
- more interesting to show a graphical update for the status of a player's health
- define a new draw function to render a visual health bar for player's shield, e.g.

```
# draw a status bar for the player's health - percentage of health
def drawStatusBar(surface, x, y, health_pct):
    # defaults for status bar dimension
    BAR_WIDTH = 100
    BAR_HEIGHT = 10
    # use health as percentage to calculate fill for status bar
    bar_fill = (health / 100) * BAR_WIDTH
    # rectangles - outline of status bar &
    bar_rect = pygame.Rect(x, y, BAR_WIDTH, BAR_HEIGHT)
    fill_rect = pygame.Rect(x, y, bar_fill, BAR_HEIGHT)
    # draw health status bar to the game window - 1 specifies pixels for border width
    pygame.draw.rect(surface, GREEN, fill_rect)
    pygame.draw.rect(surface, WHITE, bar_rect, 1)
```

- function accepts four parameters, which allow us to define
  - *a surface for rendering*
  - *its x and y location in the game window*
  - *then update the status of the player's health*
- set a default width and height for the status bar
  - *then specify how much of this bar needs to be filled with colour*
  - *colour fill relative to the player's current health status...*
- health status can be calculated as a percentage
  - *allows us to easily modify the relative sizes for the status bar*

resources

- notes = player-health-intro.pdf
- code = playerhealth1.py

Games and dramatic elements

examples of premise in games

Space Invaders

- classic example of a shoot-em up game
- simple premise for this game
  - *easy to extrapolate and apply to game's mechanics, gameplay, and challenge*
- game is set on a planet currently being attacked by advancing aliens
- game's protagonist is responsible for fighting off these aliens and saving the planet
- game will start as the aliens start advancing down the screen
  - *and the player starts firing their weapon...*

Diablo

- first released in 1996 by Blizzard Entertainment
  - *Diablo III available for latest consoles &c.*
- more detailed premise for this game
- allows the player to act out the role of a wandering warrior
- located in a town called *Tristram*
- the town has been attacked and ravaged by Diablo
- player is acting in reponse to a call of help from the people of this town
  - *who need the player to defeat Diablo and his army of the undead*
- army is located in the dungeon beneath the town's church
- game will start as the player accepts the town's proposal
- the game leads to a final confrontation with Diablo in Hell

Diablo III - console

Games and dramatic elements

characters

- as we define our game's story, and the premise for its structure, gameplay, &c.
  - *a core consideration is the nature of our game's characters*
- characters form the route, conduit, or agent for a player
  - *a player may experience the game through these characters*
- this identification becomes an important consideration for our design
  - *helps promote a sense of immersion and internalisation*
- a player will often start to empathise with a character
  - *their role in the game*
  - *their inherent need to often resolve the game's story*
- from a pyschological standpoint, a dramatic character is often perceived
  - *an extension of fears and desires often projected by a player &c.*
- such characters will often embody certain characteristics - good and bad
  - *may be associated with a greater goal or need of the player*
- a character may also be influenced by a game's type or genre
  - *often why we encounter stereotypes &c. in certain game genres, series...*
- may help lessen the need to deconstruct the game's story
  - *effectively making it easier to accept the premise of the game...*
- the *protagonist*
  - *a game's main character*
  - *often helps drive a sense of conflict and challenge*
  - *by engaging with a defined problem or series of related problems*
  - *this sense of conflict will help drive the story*
- the *antagonist*
  - *a game's counterpoint to the main character*
  - *may be another character or a feature of the game's logic*
  - *the antagonist may be used to push back against our game's protagonist*
- without this conflict and contrast
  - *a game will often lack the necessary dramatic counterpoint*
  - *any semblance of depth to the gameplay will often be lacking...*

quick exercise

> *The skies of his future began to darken*

> *Her voice is music to his ears*

> *The ballerina was a swan, gliding across the stage.*

> *A heart of stone*

Consider the following questions relative to perceived characters in the game you outlined for the previous exercise.

- for the game's protagonist, what do they want?
- what does the antagonist need?
- what are the hopes of the player for the protagonist?
- what are the fears of the player for the protagonist?

Python and Pygame - Game Example 1

fun game extras - intro

- now start to add some fun extras to the general gameplay
  - *help improve the general player experience*
- a few examples
  - *modify health status bar to better reflect health percentages*
  - *auto fire for the laser beam to continuously shoot using space bar*
  - *fun explosions for collisions*
- many more...

Python and Pygame - Game Example 1

fun game extras - update health status colours

- modify health status bar to more accurately inform player of ship's health
- common option is to simply modify colour of status bar to reflect health status
- we may use a bright colour to indicate greater health status
  - *then change it to `RED` as a warning to the player, e.g.*

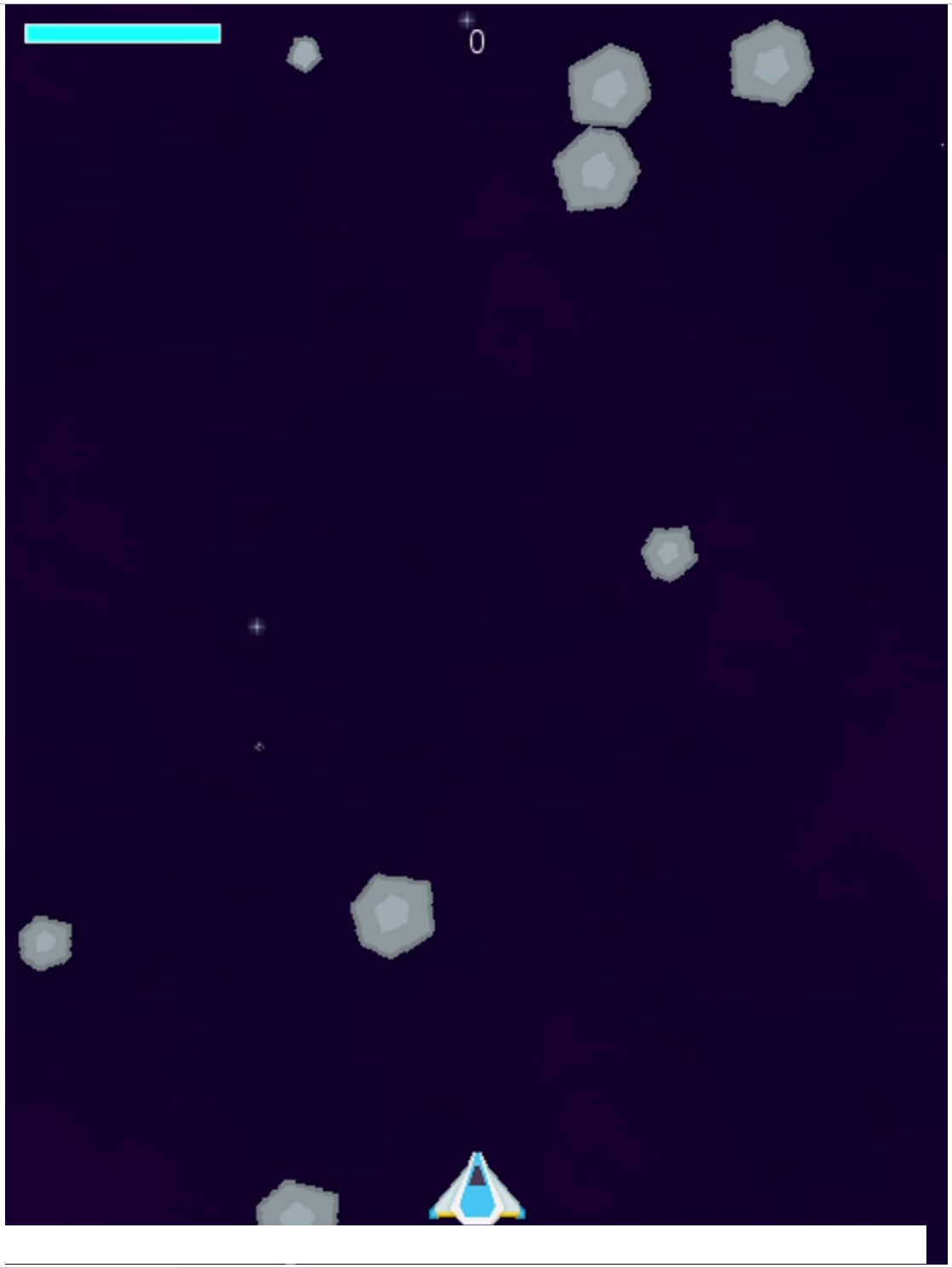
```
if bar_fill < 40:
    pygame.draw.rect(surface, RED, fill_rect)
else:
    pygame.draw.rect(surface, CYAN, fill_rect)
```

game example

- shooter1.0.py
- check player's health
  - *set default health to 100%*
  - *decrement health per collision*
    - quit game when health reaches 0
  - *draw status bar to game window*
    - green colour for good health
    - change to red colour below 40%

Video - Shooter 1.0

check player's health



Games and dramatic elements

considerations of game characters

- Characters in our games may also exhibit certain traits
  - *often unique to an interactive gaming environment*
- e.g. ability of a protagonist to become an agent in the game
  - *and channel empathy from a player to the game*
- traits of a character, in particular a game's protagonist
  - *need to be considered at each stage of a game's design and development*
- help us question motivation for a particular aspect of a game
  - *perhaps a backstory that leads to a mini-challenge for our character*
- need to consider how the character as agent enables our player to complete this mini-challenge
  - *what is the justification for including this mini-challenge in our game?*
- if we start to simply add challenges, conflict, or perhaps obstacles
  - *without a consideration of agency or motivation*
  - *a game may become disjointed and lack flow for the experience*
  - *the story, its characters, and gameplay may not make sense to the player*
- such characters need not be preconceived or developed by the game's designer
- avatars may also play a role as agent within a game
  - *e.g. in Blizzard's World of Warcraft*
- avatars will often be created, designed, and managed by a player
- players may invest a great deal of time, energy, and resources into such avatars
- agency and empathy provided by these characters
  - *will often fuel a player's gameplay and social role in a gaming environment*
  - *such empathy may be increased with greater player engagement with avatars...*

Video - World of Warcraft

avatars

- Original article - BBC News - World of Warcraft: Finding love with an online avatar
- Source - YouTube

Games and dramatic elements

characters and classes in Diablo &c.

- interesting and fun aspect of original Diablo game was use of *classes* for characters
- instead of simply providing a single option for the protagonist
  - *Diablo provided three classes*
  - *classes = Rogue, Sorcerer, and the Warrior*
- expanded to six classes for Diablo III with various expansion packs
  - *further class add-ons as well*
- each character class provides different attributes, skills, and agency for the game
- not simply a matter of providing different types of characters and skills
  - *allows different players to empathise in varying ways with the game*
- no sense of one size fits all
  - *a player is provided with different ways to enjoy and complete the game*
- choice of game agent may also introduce variant paths through the game
- a player is provided with different perspectives on the story, challenges, and general gameplay
- many other games that employ a similar option for characters
  - *e.g. Nintendo's Mario Kart selector...*

Examples

- Diablo Classes - http://diablo.wikia.com/wiki/Classes

Image - Mario Kart

select a character and kart

Nintendo's Mario Kart

Games and dramatic elements

Super Mario Kart Characters

Image	Character	Acceleration	Top Speed	Weight	Handling
	Mario	Medium	High	Medium	Medium
	Luigi	Medium	High	Medium	Medium
	Princess	Very High	Medium	Medium	Low
	Yoshi	Very High	Medium	Medium	Low
	Bowser	Low	Very High	High	Medium
	Donkey Kong Jr.	Low	Very High	High	Medium
	Koopa Troopa	High	Low	Low	High
	Toad	High	Low	Low	High

- Source - Mario Wiki

Video - Super Mario Kart

characters and gameplay

- Source - YouTube

Games and dramatic elements

characters and emergent systems

- we may introduce *emergent systems* to our gaming environment
  - *creating a sense of autonomous, generated gameplay and challenge*
- add a semblance of *free will* to our characters
  - *creates a noticeable variant to standard player control*
- a traditional character's agency
  - *may be directly influenced, monitored, and controlled by the player*
- introduction of *free will* for certain characters
  - *control limitations may no longer apply*
- AI-controlled characters or emergent systems
  - *may now start to exhibit examples of autonomous behaviour*
- potential for interesting conflict may arise as a simple result of expectations
  - *e.g. player control vs a sense of limited free will for certain characters*
- The Sims - Free Will

Video - The Sims 4

Free Will

- Source - Sims 4: The Free Will Experiment - YouTube

quick exercise

Consider the following game characters and objects,

- a medieval knight
  - *carries a sword, may ride a horse, fighting skills, finite health...*
- a squire
  - *attends to the knight*
- a semi-intelligent/aware mob object - e.g. an ogre
  - *carries a club, may ride horse-like animal, fighting skills, renewable health...*
- a series of huts, caves &c. in the gaming world

Each of these characters or objects may be pre-defined or created with a sense of free will.

Define the following,

- rules for each character and object
- a brief outline for a game with these characters and objects

Then consider the following,

- how might free will affect the rules and outline for your initial game?
- what type of unexpected glitches, interactions, and features may result due to free will in this game?

Python and Pygame - Game Example 1

fun game extras - repetitive firing sequence - intro

- add a repetitive firing sequence for the player's sprite object
- in our current game logic
  - *as a player presses down on the space bar a laser beam will be fired from the top of the player's ship*
  - *one press is equal to one firing sequence...*
- to add a repetitive firing sequence
  - *need to still check that the spacebar has been pressed down*
  - *but now continue to fire a laser beam until the key is released*
- in our `Player` class we can add some new variables, e.g.
  - *specify the delay in milliseconds between each firing of the laser beam*
  - *check the time, the number of ticks, since the last beam was fired*
  - *e.g. update `Player` class as follows,*

```
...
# firing delay between laser beams
self.firing_delay = 200
# time in ms since last fired
self.last_fired = pygame.time.get_ticks()
```

fun game extras - repetitive firing sequence - fire - part 1

- add a listener for the space bar event to the `update()` method in the `Player` class

```
# check space bar for firing projectile
if key_state[pygame.K_SPACE]:
    # fire laser beam
    self.fire()
```

- update our `fire()` method to reflect this repetitive firring sequence, e.g.

```
...
# get current time
time_now = pygame.time.get_ticks()
if time_now - self.last_fired > self.firing_delay:
    self.last_fired = time_now
    ...
```

Python and Pygame - Game Example 1

fun game extras - repetitive firing sequence - fire - part 2

- our `fire()` method has now been updated as follows,

```
# fire projectile from top of player sprite object
def fire(self):
    # get current time
    time_now = pygame.time.get_ticks()
    if time_now - self.last_fired > self.firing_delay:
        self.last_fired = time_now
        # set position of projectile relative to player's object rect for centerx and top
        projectile = Projectile(self.rect.centerx, self.rect.top)
        # add projectile to game sprites group
        game_sprites.add(projectile)
        # add each projectile to sprite group for all projectiles
        projectiles.add(projectile)
        # play laser beam sound effect
        laser_effect.play()
```

- remove listener for a space bar event in the events section of the game loop

resources

- notes = extras-part1-firing.pdf
- code = repetitivefiring.py

game example

- shooter1.1.py
- add repetitive firing sequence for player's laser beam
  - *move keypress check for space bar to player class*
  - *fire laser beam whilst space pressed down*
  - *set interval in ms for firing sequence*
  - *check time between now and last firing*

Video - Shooter 1.1

add repetitive firing sequence...

Game designers

Designer example - Jenova Chen

- Jenova Chen is a Chinese game designer and director, now based in Los Angeles, USA
- after creating an experimental game called *Cloud* with Kellee Santiago
  - *whilst a student at USC's Interactive Media Division*
  - *Chen briefly worked on Spore*
- Chen is best known for games such as
  - *Cloud, flOw, Flower*
  - *and most recently Journey*
- co-founded *ThatGameCompany* with Kellee Santiago
- landed a three game deal with Sony, which included *flOw*, *Flower*, and *Journey*
  - *games exclusive to PlayStation consoles*
- his games are known for experimental use of narrative and structure
  - *and attempts to simply push what we perceive as a game...*
  - *e.g. his development of Cloud as a student*
- his collaboration with Austin Wintory on the music for *flOw* and *Journey*
  - *represents a desire and commitment to integrate various dramatic elements*
  - *music, sound effects, shapes, colour &c. into the overall gaming experience*
- underlying trend and theme to the design of his games
  - *tries to make games that don't fit cultural preconceptions*
  - *interested in sparking universal emotions and feelings beyond culture...*

Resources

- Cloud
- flOw
- Flower
- Journey
- Journey - Wikipedia
- ThatGameCompany

Image - Journey

- Source - ThatGameCompany

Games and dramatic elements

games and narrative structure

- **traditional drama** perceives the following categories as useful options for conflict
  - *a single character vs another single character*
  - *a single character vs their environment*
    - a character battling the forces of nature &c.
  - *a single character vs a machine*
    - many examples in movies...
  - *a single character vs their own inner demons*
    - a consideration of experience, morals, insanity &c.
  - *a single character vs perceptions of fate*
    - something is inevitable, bound to happen, can't be changed &c.
- a **game** will employ similar categories for its players, in particular the protagonist
  - *a single player vs another single player*
  - *a single player vs the game*
  - *and so on...*
- as these categories are played out in our games
  - *the sense of conflict they create will usually follow a discernible pattern*
  - *this pattern will escalate to a final resolution*
- escalating conflict will create a sense of tension in the gameplay
  - *usually matched and reflected in the story*
- gameplay may respond to the story, including corresponding elements
  - *such as music, visuals, speed, and a sense of risk*
- this tension will also tend to get worse, or more dramatic
  - *before it is resolved and gets better*
- this forms a classic **narrative structure** or **narative arc**
  - *it becomes a useful tool for storytelling in games*
- forms the framework and support for all dramatic media
  - ***games*** *are not excluded...*

Resources

Demos

- pygame health and status
- playerhealth1.py
- playerhealth2.py
- pygame - fun game extras
- repetitivefiring.py
- pygame - Game 1 Example
- shooter1.0.py
- shooter1.1.py

Games

- Diablo - Wikipedia
- Diablo III - console
- Super Mario Kart - Mario Wiki
- Super Mario Kart Characters - Wikipedia
- Journey - ThatGameCompany
- Journey - PS3
- Journey - Wikipedia
- World of Warcraft

Game notes

- Pygame
- player-health-intro.pdf
- extras-part1-firing.pdf

References

- Bogost, I. *Persuasive Games: The Expressive Power of Videogames*. MIT Press. Cambridge, MA. 2007.
- Bogost, I. *Unit Operations: An Approach to Videogame Criticism*. MIT Press. Cambridge, MA. 2006.
- Various
- BBC News - World of Warcraft: Finding love with an online avatar
- Dubspot - Electronic Music Production and DJ School
- The Sims - Free Will
- ThatGameCompany - Hiring

Videos

- World of Warcraft - Finding love with an online avatar - YouTube
- Super Mario Kart Characters - YouTube