Comp 388/488 - Game Design and Development

Spring Semester 2019 - Week 11

Dr Nick Hayward

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

example

- Diablo III - console

quick exercise

Consider a game's story using a combination of the following metaphors,

> *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*

Any combination of the above is permitted.

- approx. 10 minutes...

Python and Pygame - Game Example I

fun game extras - load explosion images

- need to be able to define and load our images for the explosions
  - *use a list for these images*
  - *then cycle through these explosions as required...*
- our first exanple will use a list to simply load these explosion images
  - *initially use a `for` loop to iterate over this directory and load our images, e.g.*

```
# explosions
explosion_imgs = []

# iterate over explosion images in directory
for i in range(9):
    file = 'explosion{}.png'.format(i)
    expl_img = pygame.image.load(os.path.join(img_dir, file)).convert()
    expl_img.set_colorkey(BLACK)
    explosion_imgs.append(expl_img)
```

- use built-in function, `format()`, to specify abstracted value for iterator index
  - *in this example abstracted for the required filename*
- create our image for the Pygame window
  - *set colour key to black to create our transparency for the containing shape's background*
- then append these images to our list for explosions

Python and Pygame - Game Example 1

fun game extras - create explosion sprite object - part 1

- create a new class to help us represent and organise our sprite object for *explosions*
- add a new class for this object
  - *then start by initialising this sprite, e.g.*

```
# create a generic explosion sprite - use for asteroids, player explosions &c.
class Explosion(pygame.sprite.Sprite):
    # initialise sprite
    def __init__(self, center):
        pygame.sprite.Sprite.__init__(self)
        ...
```

- after initialising this new sprite object
  - *set starting image for our explosions*
  - *set to first index position of our list for explosion images*
- need to add the rectangle for this image
  - *set its centre to the specified value of the passed parameter*
- also set initial frame for our animation
  - *we can set it to a starting default of 0*

Python and Pygame - Game Example I

fun game extras - create explosion sprite object - part 2

- animation needs to be steady and constant
  - *may create a steady framerate for the animation itself*
  - *now check the time in ticks for the last update*
- then set a default framerate for this animation
  - *modify framerate of animation to suit game requirements*

```
# create a generic explosion sprite - use for asteroids, player explosions &c.
class Explosion(pygame.sprite.Sprite):
    # initialise sprite
    def __init__(self, center):
        pygame.sprite.Sprite.__init__(self)
        # specify image for explosion sprite
        self.image = explosion_imgs[0]
        # set rect for image
        self.rect = self.image.get_rect()
        self.rect.center = center
        # set initial frame for animation
        self.frame = 0
        # check last update to animation
        self.last_update = pygame.time.get_ticks()
        # set framerate delay between animation frames - sets speed for explosion
        self.frame_rate = 50
```

Python and Pygame - Game Example 1

fun game extras - create explosion sprite object - part 3

- need to add an `update` function to our class
  - *updates image of explosion for this sprite object as time progresses*
  - *i.e. as the framerate advances, switch explosion images to create animation*

```
...
# change image as time progresses for explosion sprite
def update(self):
    # get current time
    now = pygame.time.get_ticks()
    # check if enough time has passed between animations
    if now - self.last_update > self.frame_rate:
        self.last_update = now
        # if enough time passed - add 1 to frame
        self.frame += 1
        # check if end of explosion images reached
        if self.frame == len(explosion_imgs):
            # kill if end of image reached
            self.kill()
        else:
            center = self.rect.center
            self.image = explosion_imgs[self.frame]
            # update rect for image
            self.rect = self.image.get_rect()
            self.rect.center = center
```

- need to check the current time in the game
  - *check if enough time has passed between each animation*
- if enough time has elapsed
  - *update the value for the `last_update` time record*
  - *advance our animation frame by an increment of 1*
- then `kill()` animation at the end of the explosion images...

Python and Pygame - Game Example 1

fun game extras - add explosions to collisions

- our sprite object for explosions has now been created
- now call this explosion whenever we record a collision between
  - *a projectile and a mob object*
  - *a mob object and player object...*
- in our game loop `update` section
  - *check for collisions we can now add an animation for the explosions*

```
...
# add more mobs for those hit and deleted by projectiles
for collision in collisions:
    # calculate points relative to size of mob object
    game_score += 40 - collision.radius
    # play explosion sound effect for collision
    explosion_effect.play()
    # add animation for explosion images if collision
    explosion = Explosion(collision.rect.center)
    # add explosion sprite to game sprites group
    game_sprites.add(explosion)
    # create a new mob object
    createMob()
...
```

- as we're checking for collisions, we can now
  - *update game score*
  - *play a sound effect for an explosions/collision*
  - *create the animation for the explosion effect*
  - ...

resources

- notes = extras-part1-explosions.pdf
- code = objectexplosions.py

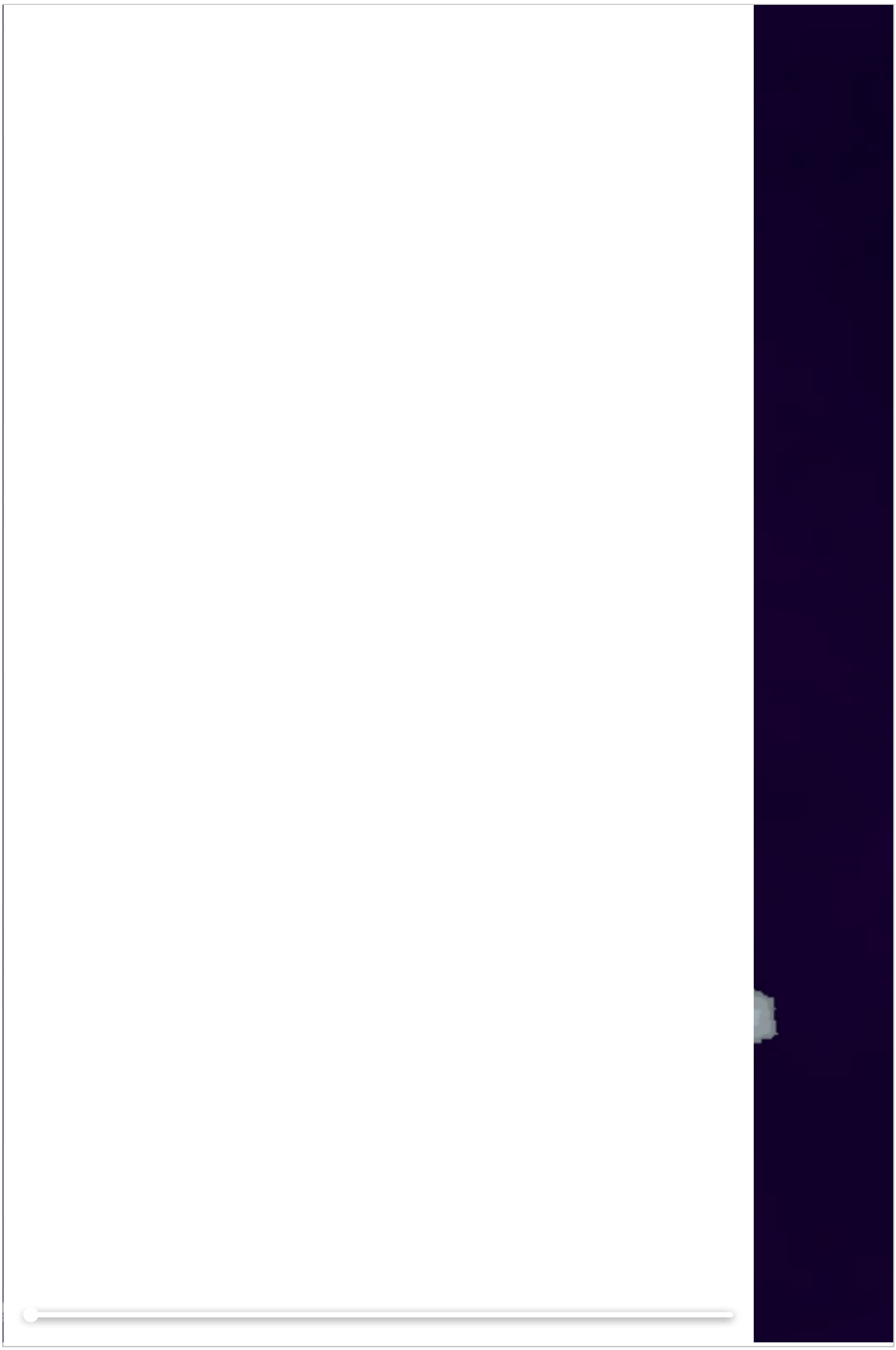
game example

- shooter1.2.py
- add some fun explosions
  - *create sprite object for explosion*

- *cycle through images to create explosion animation*
- *add explosion for each collision*

Video - Shooter 1.2 - Part 1

add some fun explosions - mob objects



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...*

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...*

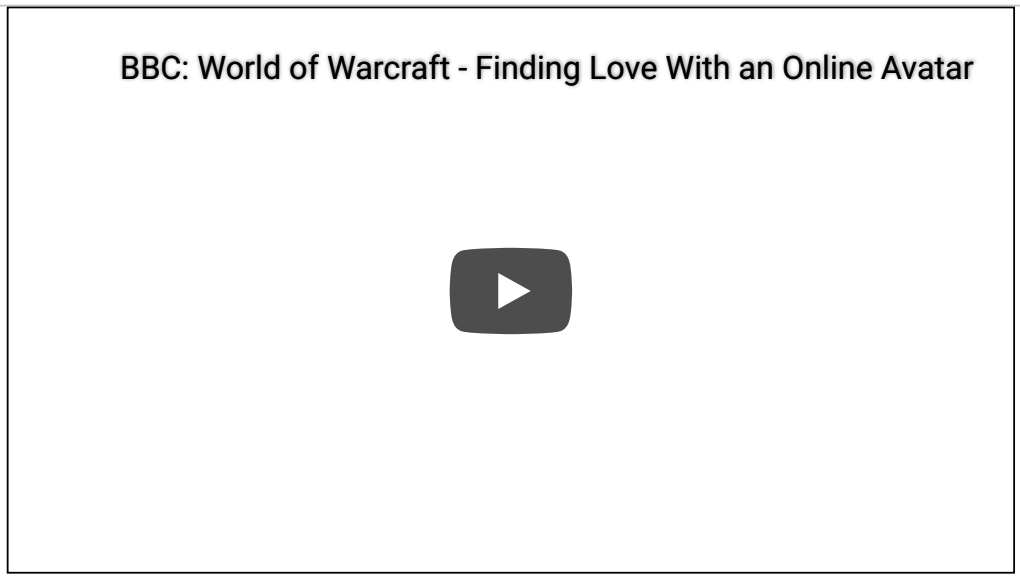
quick exercise

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?

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

- 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
  - *expected to increase to seven in 2017*
- 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...*

Image - Mario Kart

select a character and kart

Nintendo's Mario Kart

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 - The Sims 4: The Free Will Experiment - YouTube

Python and Pygame - Game Example I

fun game extras - add explosions to player's ship

- add explosions to a collision with a player's ship
  - *again, update game loop for these collisions*

```
# add check for collision - enemy and player sprites (True = hit object is now deleted fro
collisions = pygame.sprite.spritecollide(player, mob_sprites, True, pygame.sprite.collide_
# 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
    # add animation for explosion images if collision
    explosion = Explosion(collision.rect.center)
    # add explosion sprite to game sprites group
    game_sprites.add(explosion)
    # create a new mob object
    createMob()
    # check overall shield value - quit game if no shield
    if player.stShield <= 0:
        running = False
```

Video - Shooter 1.2 - Part 2

add some fun explosions - player's ship

Python and Pygame - Game Example I

fun game extras - scale explosion images - basic scale

- still a lingering issue with these collisions and explosions...
- explosions are not reinforcing the gameplay for our shooter style game
  - *no differentiation in the relative size of an explosion*
  - *no semblance of feedback to our player*
- one option to this issue
  - *perhaps add standard scale transform to image for each explosion sprite object*

```
# explosions
explosion_imgs = []

# iterate over explosion images in directory
for i in range(9):
    file = 'explosion{}.png'.format(i)
    # load image from os
    expl_img = pygame.image.load(os.path.join(img_dir, file)).convert()
    # set colour key for image
    expl_img.set_colorkey(BLACK)
    # append to specified list for explosion images
    explosion_imgs.append(expl_img)
```

- render a smaller, less overwhelming explosion for each collision

Python and Pygame - Game Example 1

fun game extras - scale explosion images - relative scale - part 1

- useful to be able to scale these explosions relative to the actual size of a given sprite object
  - *e.g. a smaller relative explosion image for a smaller mob object*
  - *or, a relatively sized explosion against the player's ship*
- update our class for the `Explosion` object
  - *dynamically modify each explosion image in the animation relative to a specified size*
- scale each frame of explosion animation to match the size of the collison object, e.g.

```
# create a generic explosion sprite - use for asteroids, player explosions &c.
class Explosion(pygame.sprite.Sprite):
    # initialise sprite
    def __init__(self, center, size):
        pygame.sprite.Sprite.__init__(self)
        # specify size for explosion sprite
        self.size = size
        # get initial image for explosion
        self.image = pygame.transform.scale(explosion_imgs[0], self.size)
...
```

- start by adding a parameter for `size`
  - *pass a variable size for each collision object*
- use this size to scale the initial image for the explosion animation

Python and Pygame - Game Example 1

fun game extras - scale explosion images - relative scale - part 2

- each frame of the animation will also require scaling of the explosion image, e.g.

```
# change image as time progresses for explosion sprite
def update(self):
    # get current time
    now = pygame.time.get_ticks()
    # check if enough time has passed between animations
    if now - self.last_update > self.frame_rate:
        self.last_update = now
        # if enough time passed - add 1 to frame
        self.frame += 1
        # check if end of explosion images reached
        if self.frame == len(explosion_imgs):
            # kill if end of image reached
            self.kill()
        else:
            center = self.rect.center
            self.image = pygame.transform.scale(explosion_imgs[self.frame], self.size)
            # update rect for image
            self.rect = self.image.get_rect()
            self.rect.center = center
```

- as we output each frame of the explosion animation
  - *scale this image to match the passed `size` for the explosion object*

Python and Pygame - Game Example 1

fun game extras - scale explosion images - dynamic collision size

- different size mob objects will have a matching explosion animation
  - *update in the game loop, e.g.*

```
# add check for sprite group collide with another sprite group - projectiles hitting enemy
collisions = pygame.sprite.groupcollide(mob_sprites, projectiles, True, True)
# add more mobs for those hit and deleted by projectiles
for collision in collisions:
    # calculate points relative to size of mob object
    game_score += 40 - collision.radius
    # play explosion sound effect for collision
    explosion_effect.play()
    # get size of collision object
    col_size = collision.rect.size
    #print("collision size = " + str(col_size))
    # add animation for explosion images if collision
    explosion = Explosion(collision.rect.center, col_size)
    # add explosion sprite to game sprites group
    game_sprites.add(explosion)
    # create a new mob object
    createMob()
```

- same for the player's object...

resources

- notes = extras-part1-explosions.pdf
- code = objectexplosions2.py

game example

- shooter1.2.py
- add some fun explosions
  - *create sprite object for explosion*
  - *cycle through images to create explosion animation*
  - *add explosion for each collision*
- extra explosions
  - *explode a player's ship for a collision*
- scale explosions
  - *rescale and size explosions in game window*

Video - Shooter 1.2 - Part 3

scale explosions

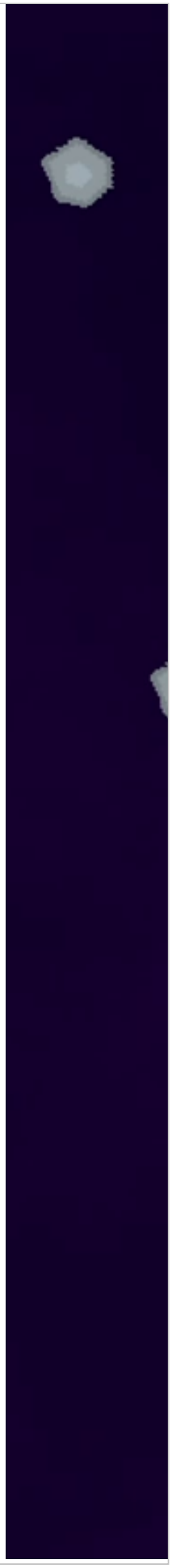


Image - Journey

- Source - ThatGameCompany

Demos

- pygame - fun game extras
  - *repetitivefiring.py*
  - *objectexplosions1.py*
  - *objectexplosions2.py*
- pygame - Game 1 Example
  - *shooter1.2.py*

Games

- Diablo - Wikipedia
- Diablo III - console
- World of Warcraft

Game notes

- Pygame
  - *extras-part1-explosions.pdf*
- Game examples
  - *Shooter.pdf*

References

- Bogost, I. *Persuasive Games: The Expressive Power of Videogames*. MIT Press. Cambridge, MA. 2007.
- Bogost, I, *The Rhetoric of Video Games*. in *The Ecology of Games*... Salen, E. MIT Press. Cambridge, MA. 2008.
- Bogost, I. *Unit Operations: An Approach to Videogame Criticism*. MIT Press. Cambridge, MA. 2006.
- Csikszentmihalyi, M. *Flow: The Psychology of Optimal Experience*. Harper & Row. New York. 1990.
- Huizinga, J. *Homo Ludens: A Study of the Play-Element in Culture*. Angelico Press. 2016.
- Murray, J. *Hamlet on the Holodeck: The Future of Narrative in Cyberspace*. Free Press. New York. 1997.
- Poundstone, W. *Prisoner's Dilemma*. Touchstone. New York. 2002.
- Salen, K. & Zimmerman, E. *Rules of Play: Game Design Fundamentals*. MIT Press. 2003.
- 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*