Comp 388/488 - Game Design and Development

Spring Semester 2018 - Slides - week 6

Dr Nick Hayward

Games and Ideas

express ideas in video games - part 1

- often begin game development by representing behaviour and structure of real-world system
  - *e.g. cars driving, people walking, planes flying...*
  - *such systems are apparent throughout our games*
- begin building our game
  - *usually start with a known model of our chosen system*
  - *also coding potential outcomes*
  - *one of the inherent features of coding and development*
- such outcomes are developed to meet the defined requirements for a set of rules
  - *usually those defined for the system itself*
  - *or combined with the rules of the game*
- J. Murray, in 1997
  - *referred to this simply as a* ***procedural representation***
  - *video games are good at this type of representation*
- classic example of such procedural representation is the popular game *Sim City*
  - *models urban development, planning, general dynamics of city and urban living...*
  - *able to model societal and cultural patterns within this urban environment*
    - e.g. crime rates, pollution levels, economy...
- Ian Bogost explains that

> *"video games represent processes in the material world - war, urban planning, sports, and so forth- and create new possibility spaces for exploring those topics."*

Bogost, I, *The Rhetoric of Video Games*. in *The Ecology of Games*... Salen, E. MIT Press. Cambridge, MA. 2008.

Games and Ideas

express ideas in video games - part 2

- as we begin development of our game
  - *we are expressing ideas of a given system*
  - *often in a procedural manner*
- as our players experience the game
  - *they begin to form an impression or idea of the system itself*
  - *the underlying system being represented*
- the game has started to impart its ideas upon the player
- designers and developers represent their own interpretations and impressions
  - *of the underlying real-world system in the game*
- does this system actually exist in the first place?
  - *Bogost, I. has argued such video game systems inherently speculative*
  - *derived from the developer, not directly from the system itself*
- such subjectivity naturally creates a tension and dissonance, according to Bogost, I.
  - *between the player's pre-conceptions of a system*
  - *and the developer's implementation*
- tension helps express the game itself, encouraging a player to
  - *explore*
  - *question*
  - *and test the game's own systems, concepts, and general gameplay*
- can be a valuable reason to continue playing the game

Games and Ideas

express ideas in video games - part 3

- Bogost describes models as a good form of representing procedural game play
- Sid Meir's **Civilization** series of games
  - *each game can be thought of in terms of a model*
  - *a model of how real world, perceived global affairs occur...*
- specifics of the game may use ancient history and societies its model
  - *may serve as a model of many principles governing international relations today*
  - *game processes, logical outcomes reflect known world operations*
- each game uses a procedural model
  - *a player still maintains a certain degree of agency*
- player's gameplay procedure may affect the experience
  - *to an equal extent as the game's procedure...*
- each game provides an opportunity to interpret systems, rules, and procedures
- player may decide how to interpret and modify their meaning
  - *within their gameplay and experience...*
- *Civilization* series is a great example of **procedural representation** in gaming

Video - Procedural Representation

Civilization series

Source - Sid Meier's Civilization, Youtube

Video - Procedural Representation

Animal Crossing

Source - Animal Crossing, YouTube

quick exercise

Consider the following real-world systems:

Motion

- cars and driving
- planes and flying
- human motion and interaction

Societal

- informal groups
- hierarchy and formal organisations
- family

Then,

- define the known models for at least one of these systems per group, *Motion* and *Societal*
- consider potential outcomes for your chosen systems
- consider how a game may then use such systems and outcomes in a procedural representation
- consider how your game may then modify and push such systems and outcomes to create a sense of *play*

Video - a fun example

The Last Starfighter - Theatrical Trailer

Source - The Last Startfighter, YouTube

Game Dev resources

gamedev.net

- game dev resources - various updates, links, suggestions...
- a long standing example - **gamedev.net**
  - *https://www.gamedev.net/*
- original founded in 1999
  - *great resource for general game development*

Fun gaming music covers

- Gaming music playlist 1
  - *Lindsey Stirling - Various Gaming Music Videos*
- Gaming music playlist 2
  - *Taylor Davis - Video Game Covers*
- covers include:
  - *Dragon Age, Halo, Zelda, Skyrim, Assassin's Creed, Mass Effect, The Witcher...*

Fun gaming inspirational music

- Really Slow Motion - YouTube Channel

Game Dev and Design - Extras

Graduate Courses

A few example game design and development courses:

- New York University - Game Center
  - *more design oriented*
- University of Southern California - USC Games
  - *highest ranked school in many game design degree tables...*
  - *four applicable degree programmes - 2 Graduate*
  - *good connections with industry...*
- University of Utah - Entertainment Arts & Engineering
  - *good reputation for hands-on design and development*
  - *a good mix of design and development - cross-tracks...*
  - *close links to industry - e.g. EA*
- New York Film Academy - Game Design and Development School

Lots of options at the following URL,

- Video Game Design Schools

Python and Pygame - Game Example I

better collison detection

- collision detection is currently using rectangles to detect one sprite colliding with another
  - *technically referred to as **Axis-aligned Bounding Box** (AABB)*
- for some sprite images this will often cause an unrealistic effect as the two images collide
  - *image does not appear to collide with the other image*
  - *due to space caused by each respective rectangle*
- as one corner of a rectangle hits another corner a collision will be detected, e.g.

```
 ________
|        |
|        |
|       _|_____
|______|_|     |
       |       |
       |_______|
```

- unless each sprites image fits exactly inside the respective bounding box
  - *there will be space left over...*
- a few options for rectifying this issue
  - *might choose to simply calculate and set a slightly smaller rectangle*
  - *or, use a circular bounding box for our sprite image*
- benefit of an *axis-aligned bounding box*
  - *game is able to detect and calculate collisions much faster for a rectangle bounding box*
- a *circular bounding box* may be slower
  - *simply due to the number of calculations the game may need to perform*
  - *checks radius of one bounding box against another bounding box radius*
- rarely becomes a practical issue
  - *unless you're trying to work with thousands of potential sprite images*
- another option, the most precise as well
  - *use pixel perfect collision detection (PPCD)*
- PPCD - game engine will check each pixel of each possible sprite image
  - *determines if and when they collided*
  - *particularly resource intensive unless you require such precision*

Python and Pygame - Game Example 1

add circle bounding box - part 1

- add some *circle bounding boxes* to our sprite images
  - *for `player` and `mob` objects*
- start by adding explicit circles with a fill colour
  - *helps us check the relative position of the circle's bounding box, e.g.*

```
self.radius = 20
pygame.draw.circle(self.image, RED, self.rect.center, self.radius)
```

- we know sprite image for player's object will have a fixed, known size
  - *we may set the radius to `20`*
- we may add some *circle bounding boxes* to the mob objects as well, e.g.

```
self.radius = int(self.rect.width * 0.9 / 2)
pygame.draw.circle(self.image, RED, self.rect.center, self.radius)
```

Python and Pygame - Game Example I

add circle bounding box - part 2

- used same basic pattern to add circles
  - *for mob objects we may set each circle's radius relative to the sprite image*
  - *setting radius as 90% of width of sprite image*
  - *then returning half of that value...*
- to use each *circle bounding box*, we need to update the collision checks as well
  - *update this check for each mob object in the update section of the game loop, e.g.*

```
# add check for collision - enemy and player sprites (False = hit object is not deleted from game window)
collisions = pygame.sprite.spritecollide(player, mob_sprites, False, pygame.sprite.collide_circle)
```

- updated the collision check to explicitly look for circle collisions
  - *now remove explicit drawn circle for each circle bounding box*
  - *e.g. for the player and mob object sprites*
  - *we may simply comment out the drawn circle*

```
self.radius = int(self.rect.width * 0.9 / 2)
#pygame.draw.circle(self.image, RED, self.rect.center, self.radius)
```

resources

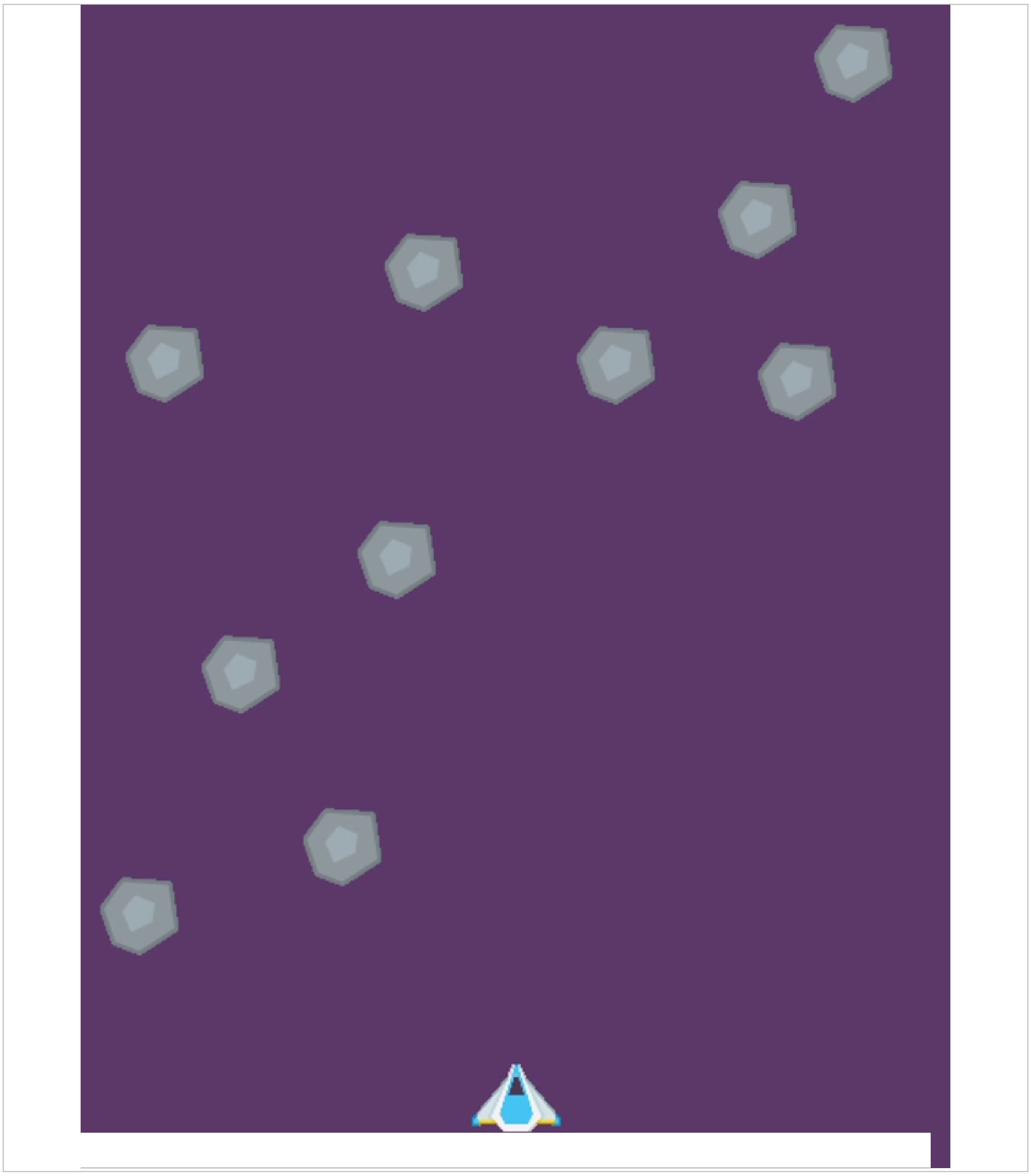
- notes = sprites-collision-detection-better.pdf
- code = collisionsprites3.py

game example

- shooter0.5.py
  - *better collisions and detection*
    - change bounding box for player and mob sprite objects
    - change bounding box to circle, and modify radius to fit sprite objects

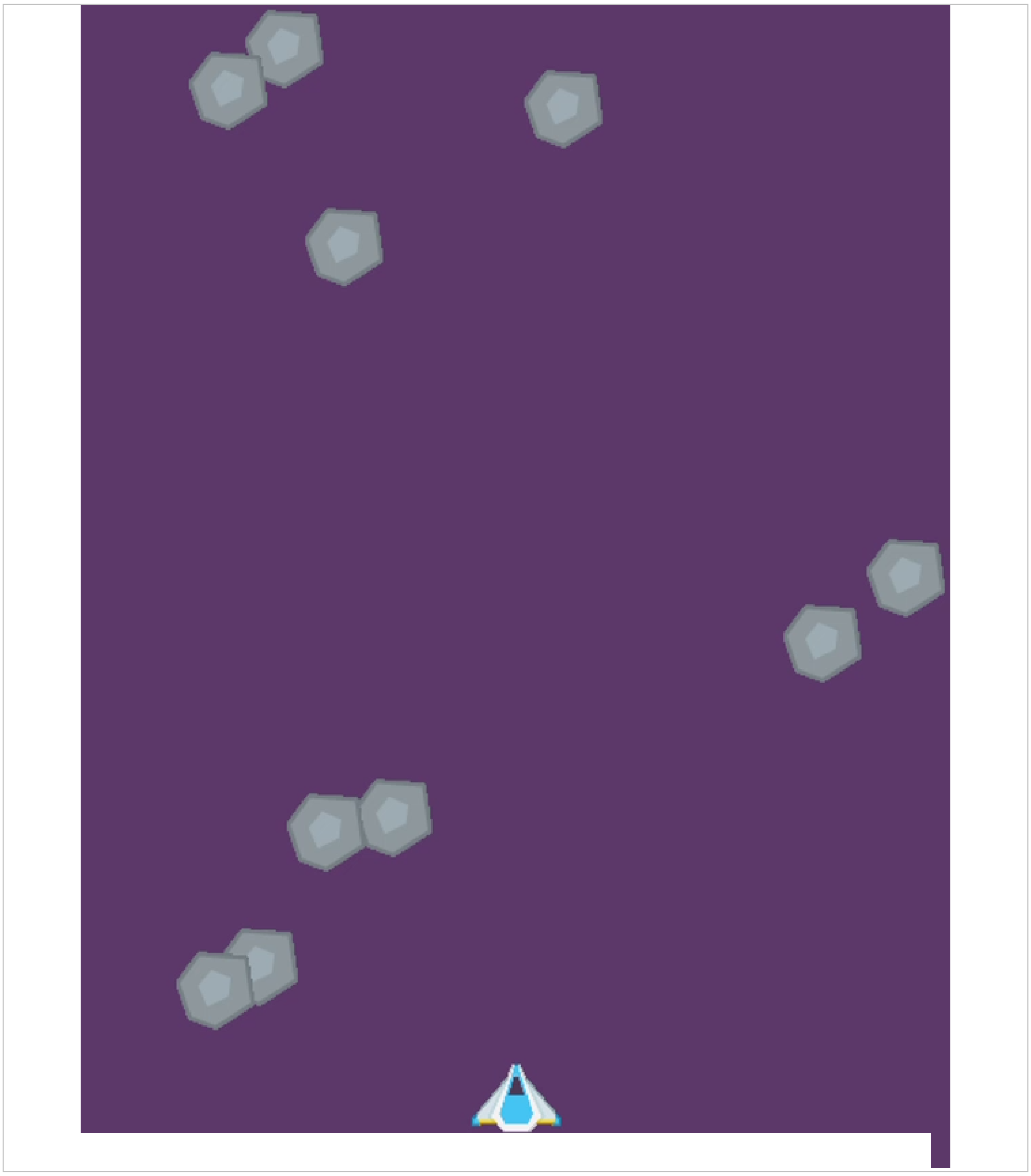
Video - Shooter0.5

better collision detection - example 1



Video - Shooter0.5

better collision detection - example 2



Intro - Collision Detection

A bit of fun - Atari Asteroids

A brief intro from Wikipedia,

> *Asteroids is an arcade space shooter released in November 1979 by Atari, Inc. and designed by Lyle Rains, Ed Logg, and Dominic Walsh. The player controls a spaceship in an asteroid field which is periodically traversed by flying saucers. The object of the game is to shoot and destroy asteroids and saucers while not colliding with either or being hit by the saucers' counter-fire. The game becomes harder as the number of asteroids increases.*

> *Asteroids was one of the first major hits of the golden age of arcade games. The game sold over 70,000 arcade cabinets and proved both popular with players and influential with developers. It has since been ported to multiple platforms. Asteroids was widely imitated and directly influenced Defender, Gravitar, and many other video games.*

I need some volunteers

- Atari Asteroids - Wikipedia entry
- Atari Asteroids - Play Online
- Atari Arcade

Game Dev resources

music, sound effects...

- for a game's sound effects
  - *many different options and sources for these sounds*
- try open source examples, e.g.
  - *Open Game Art*
- perhaps create our own custom sounds using a utility such as **SFXR**, e.g.
  - *SFXR*
- or its derivative website option, e.g.
  - *BFXR*

Games and formal elements

intro

- as with each formal structure
  - *players*
  - *objectives*
  - *procedures & rules*
    - including implied **boundaries**
  - *conflict, challenge, battle...*
  - *outcome, end result...*
- these constituent elements come together
  - *to form what we largely understand to be a game*
- such formal elements constitute how we
  - *design*
  - *structure*
  - *develop our video games*
- overlap and interplay of these formal elements
  - *has now become the foundation for game design*
- a sound understanding and knowledge of these formal elements
  - *their usage and application*
  - *helps us start creating innovative, playful game experiences*

Games and formal elements

players and games - part 1

- identified the need for rules, procedures...
- within the confines of such rules
  - *players are suspending normal societal restrictions*
  - *players enact shooting, fighting, role-play, and magical roles...*
  - *roles, actions &c. normally confined to a passive medium, e.g. books, film...*
- such actions can often form stark contrasts in a game environment
- rules become enacted in a **magic circle**
  - *originally described by Huizinga in his 1938 title, **Homo Ludens***
  - *later adapted, and refashioned for digital games*

> *In a very basic sense, the magic circle of a game is where the game takes place. To play a game means entering into a magic circle, or perhaps creating one as a game begins.*

Salen, K. & Zimmerman, E. *Rules of Play: Game Design Fundamentals*. MIT Press. 2003.

Games and formal elements

players and games - part 2

- such rules naturally create the opportunity for play
  - *within the defined confines of a game*
- our use of rules, characters, story, and even mechanics and control
  - *invites a player to become involved with, and invested in, our game*
- motion controllers became an invitation for players to intuitively enter a game
  - *e.g. Nintendo's Wii, Microsoft's Xbox Kinect, and Sony's Playstation Move...*
- the premise of many games now became an extension of the controller
- it's not only a matter of engaging and inviting players into your game
- need to consider the nature and structure of player participation, e.g.
  - *how many players does the game support?*
  - *will each player adopt the same role?*
  - *perhaps play in a team or in direct competition*
- how we answer such questions will have a direct influence
  - *on the nature of the game we're designing*
  - *its gameplay*
  - *and a player's engagement with the story and characters...*

Image - Motion Controllers

Nintendo Wii	Xbox Kinect	Playstation Move

Video - Xbox Kinect & Algorithms

Kinect algorithm starts at 47:36 minutes into the video.

Source - Algorithms, YouTube

Games and formal elements

players, patterns, and numbers

- games may be designed and developed for a variety of player numbers
  - *from strict single player options to varied multiplayer environments*
  - *MMOG push player numbers to the upper bounds*
- player numbers will often determine patterns of interaction for your game
- patterns may include
  - *single player versus the game*
    - e.g. Space Invaders, Mario, Sonic...
  - *multiple individual players versus the game*
    - e.g. sports, racing, card games...
  - *single player versus another player*
    - e.g. games such as Street Fighter and Mortal Kombat...
  - *multiple players versus a single player*
    - many detective and role playing board games include such features
    - some *god* games may also be structured using this pattern
  - *collaborative play*
    - players work together against the game
    - sports games, *Journey* by designer Jenova Chen...
  - *multiple players competing*
    - e.g. Halo, Call of Duty...
    - standard pattern referenced for *multiplayer* games
  - *team competitions...*
    - e.g. eSports such as League of Legends...

Python and Pygame - Game Example I

animating sprite images - part 1

- for many game sprites it's fun and useful to add different animations
  - *to animate different states, actions, &c. as the game progresses...*
  - *e.g. random rotation of mob objects, explosions, collisions...*
- already added scale transform to mob objects
  - *we may use the same pattern to add a rotate option*
  - *add animation to these sprites as they move down the game window*
  - *e.g. start by setting some variables for our rotation,*

```
# set up rotation for sprite image - default rotate value, rotate speed to add diff. directions,
self.rotate = 0
self.rotate_speed = random.randrange(-7, 7)
```

- due to the framerate of this game, set to 60FPS
  - *need to ensure rotate animation does not occur for each update of the game loop*
  - *if not, rotation will be too quick, unrealistic, annoying...*

animating sprite images - part 2

- in addition to the rotate animation
  - *also need to consider how to create a timer for this animation*
  - *regularity of update to the animation to ensure it renders realistically*
- already a timer available within our existing code
  - *currently using to monitor the framerate for our game*
- use this timer to check the last time we updated our mob sprite image
- set a time to rotate the sprite image
  - *then check this monitor as it reaches this specified time*
- record last time our sprite image was rotated by getting the time
  - *number of ticks since the game started, e.g.*

```
# check timer for last update to rotate
self.rotate_update = pygame.time.get_ticks()
```

- each time the mob sprite image object is rotated
  - *update value of variable to record the last time for a rotation*
  - *modify the mob sprite's `update` function as follows,*

```
# call rotate update
self.rotate()
```

- simply going to call a separate rotate function
  - *keep the code cleaner and easier to read*
  - *allows us to quickly and easily modify, remove, and simply stop our object's rotation*

rotate

- now add our new `rotate()` function
  - *start by checking if it's time to rotate the sprite image*

```
def rotate(self):
      # check time - get time now and check if ready to rotate sprite image
      time_now = pygame.time.get_ticks()
      # check if ready to update...in milliseconds
      if time_now - self.rotate_update > 70:
          self.last_update = time_now
```

- uses the current time, relative to the game's timer
  - *then checks this value against the last value for a rotate update*
- if difference is greater than 70 milliseconds
  - *it's time to rotate the sprite object*

Python and Pygame - Game Example I

rotate issues

- for rotation we can't simply add a *rotate transform* to the `rotate()` function
  - *possible in the code, it will also cause the game window to practically freeze*
  - *makes the game unplayable in most examples, e.g.*

```
self.image = pygame.transform.rotate(self.image, self.rotate_speed)
```

- this issue is due to pixel loss for the image
- each rotation of a sprite object image
  - *causes a game's logic to lose part of the pixels for that image*
- this will cause the game loop to start to freeze...

correct rotation

- correct this rotation issue by working with an original, pristine image for the sprite object

```
# set pristine original image for sprite object
self.image_original = mob_img
# set colour key for original image
self.image_original.set_colorkey(BLACK)
```

- then, set the initial sprite object image as a copy of this original

```
# set copy image for sprite rendering
self.image = self.image_original.copy()
```

- then, we may use the pristine original image with the rotation

```
self.image = pygame.transform.rotate(self.image_original, self.rotate_speed)
```

Python and Pygame - Game Example I

correct rotation speed

- another issue we need to fix is the rotation speed for a sprite object image
- if we simply use our default `self.rotate_speed`
  - *not keeping track of how far we've actually rotated the image*
- need to keep a record of incremental rotation of the image
  - *ensure that it rotates smoothly and in a realistic manner*
- monitor this rotation by using the value of the rotation
  - *adding rotation speed for each update to a sprite object image*
- as the image rotates we can simply check its value as a modulus of 360
  - *to ensure it keeps rotating correctly*

```
self.rotate = (self.rotate + self.rotate_speed) % 360
self.image = pygame.transform.rotate(self.image_original, self.rotate)
```

Python and Pygame - Game Example 1

rect rotation issues

- still have an issue with the rectangle bounding box, which does not rotate
- as sprite image rotates, it loses its centre relative to the bounding rectangle
- to correct this issue, we can now modify our logic for the sprite object's *update*, e.g.

```
# new image for rotation
rotate_image = pygame.transform.rotate(self.image_original, self.rotate)
# check location of original centre of rect
original_centre = self.rect.center
# set image to rotate image
self.image = rotate_image
# create new rect for image
self.rect = self.image.get_rect()
self.rect.center = original_centre
```

- mob sprite object images will now correctly rotate as they move down the screen

resources

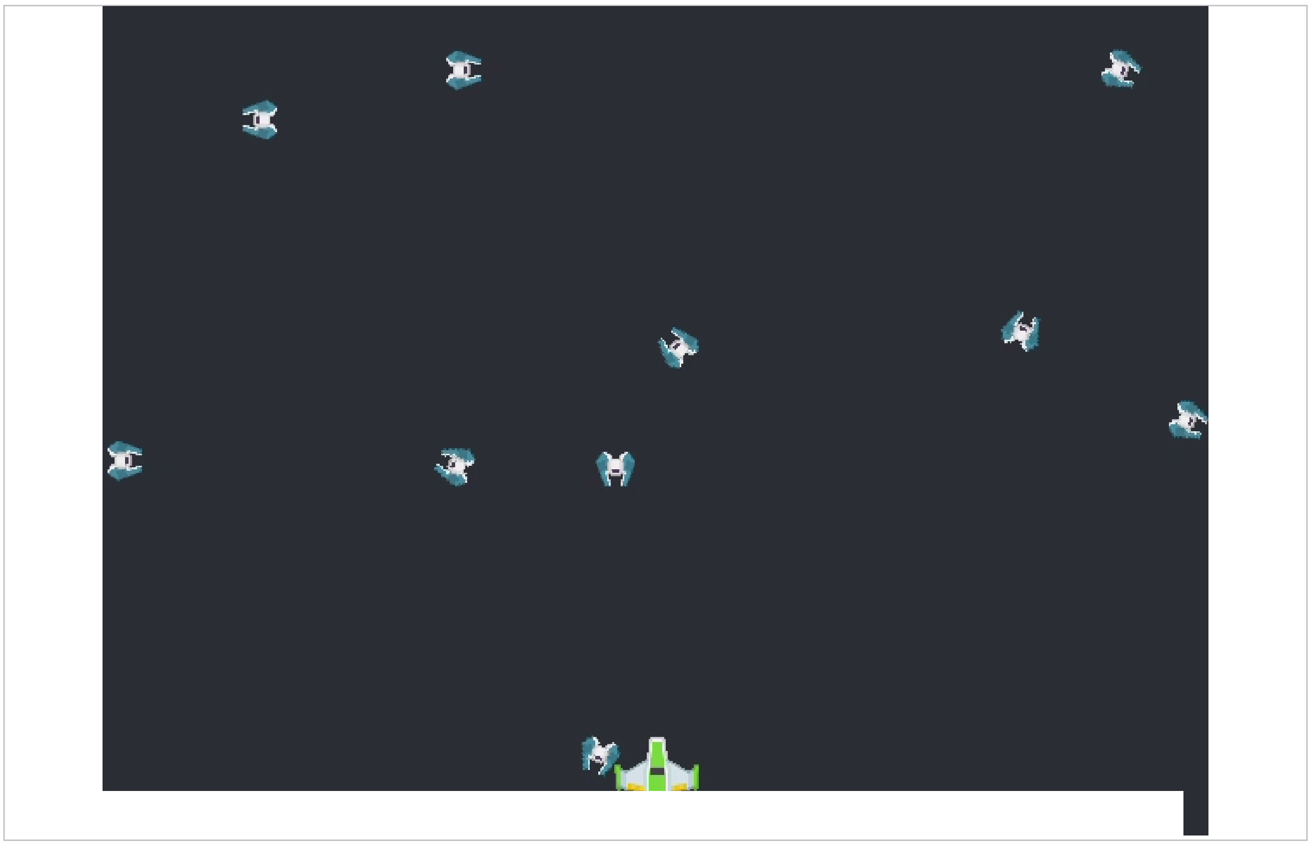
- notes = sprites-animating-images.pdf
- code = animatingsprites1.py

game example

- shooter0.6.py
  - *animating sprite images*
    - rotate mob images down the screen
    - create pristine image for rotation
    - update rect bounding box to ensure it rotates correctly

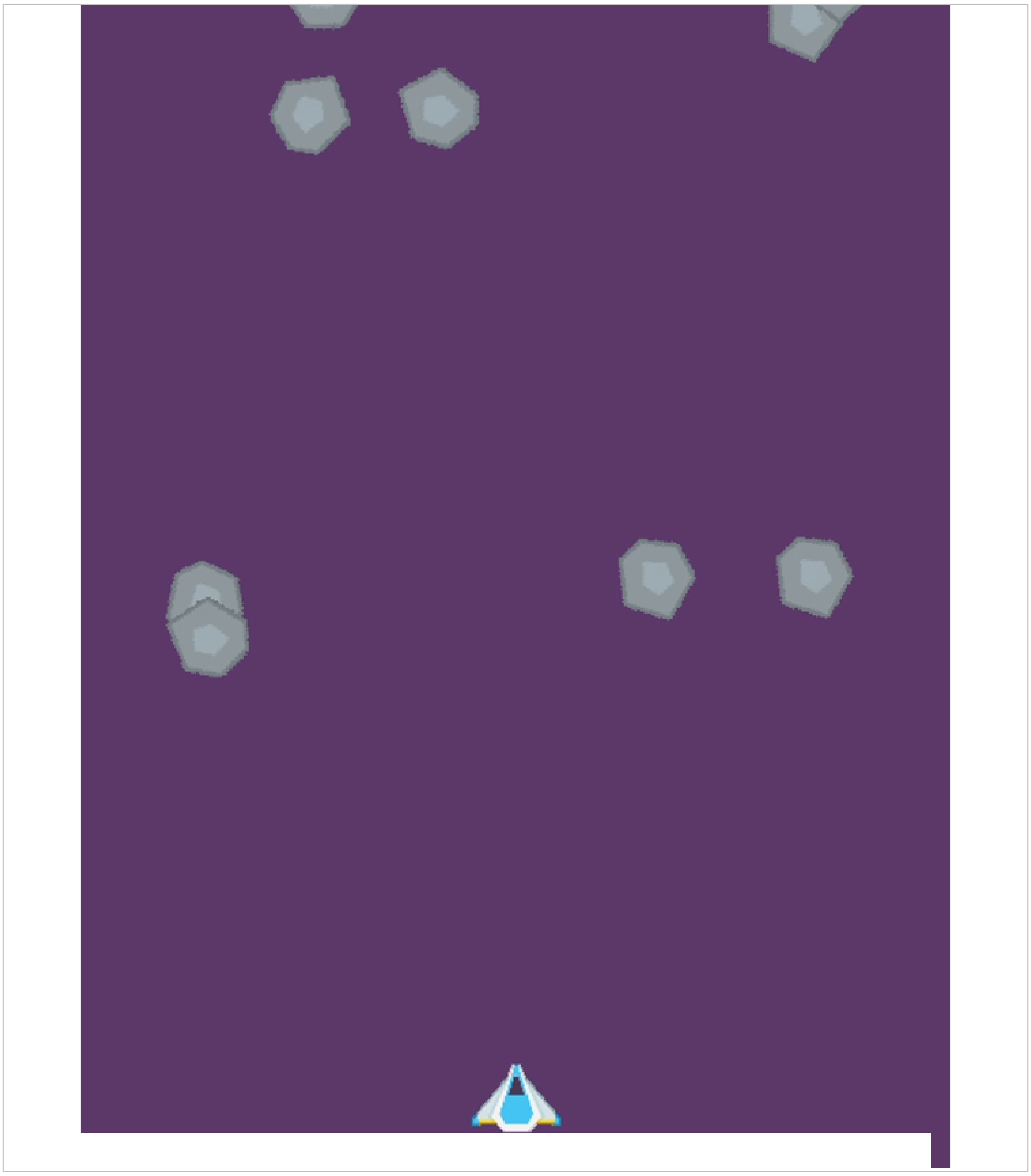
Video - animating sprites

rotation



Video - Shooter 0.6

animating sprite images



Intro - Sprites and Animation

A bit of fun - Bouncing Balls

A brief intro from Wikipedia,

> *The physics of a bouncing ball concerns the physical behaviour of bouncing balls, particularly its motion before, during, and after impact against the surface of another body. The motion of a ball is generally described by projectile motion (which can be affected by gravity, drag, the Magnus effect, and buoyancy), while its impact is usually characterized through the coefficient of restitution (which can be affected by the nature of the ball, the nature of the impacting surface, the impact velocity, rotation, and local conditions such as temperature and pressure).*

I need some volunteers

- Physics of a Bouncing Ball - Wikipedia entry
- Bouncing Balls - Play Online

Python and Pygame - Game Example I

random mob sprite images

- as we add sprite image objects to a game window, e.g. multiple *mob* images
  - *we can make the game experience more fun*
  - *by randomising the image for each mob sprite object*
- we may use a group of images as possible mob images
  - *then randomise their selection for each new mob sprite object image*
- to add random image, at least randomised from potential options...
  - *need to add a list of available images for the random selection, e.g.*

```
asteroid_list = ["asteroid-tiny-grey.png", "asteroid-small-grey.png", "asteroid-med-grey.png"]
```

- also need a new list for our asteroid images, e.g.

```
asteroid_imgs = []
```

- simply need to loop through this asteroid list
  - *then add each available image to the list of `asteroid_imgs`, e.g.*

```
for img in asteroid_list:
    asteroid_imgs.append(pygame.image.load(os.path.join(img_dir, img)).convert())
```

- then update the `Mob` class to set a random image from `asteroid_imgs` list, e.g.

```
self.image_original = random.choice(asteroid_imgs)
```

- images for our mob sprite objects will now be randomly chosen from the available list of images

resources

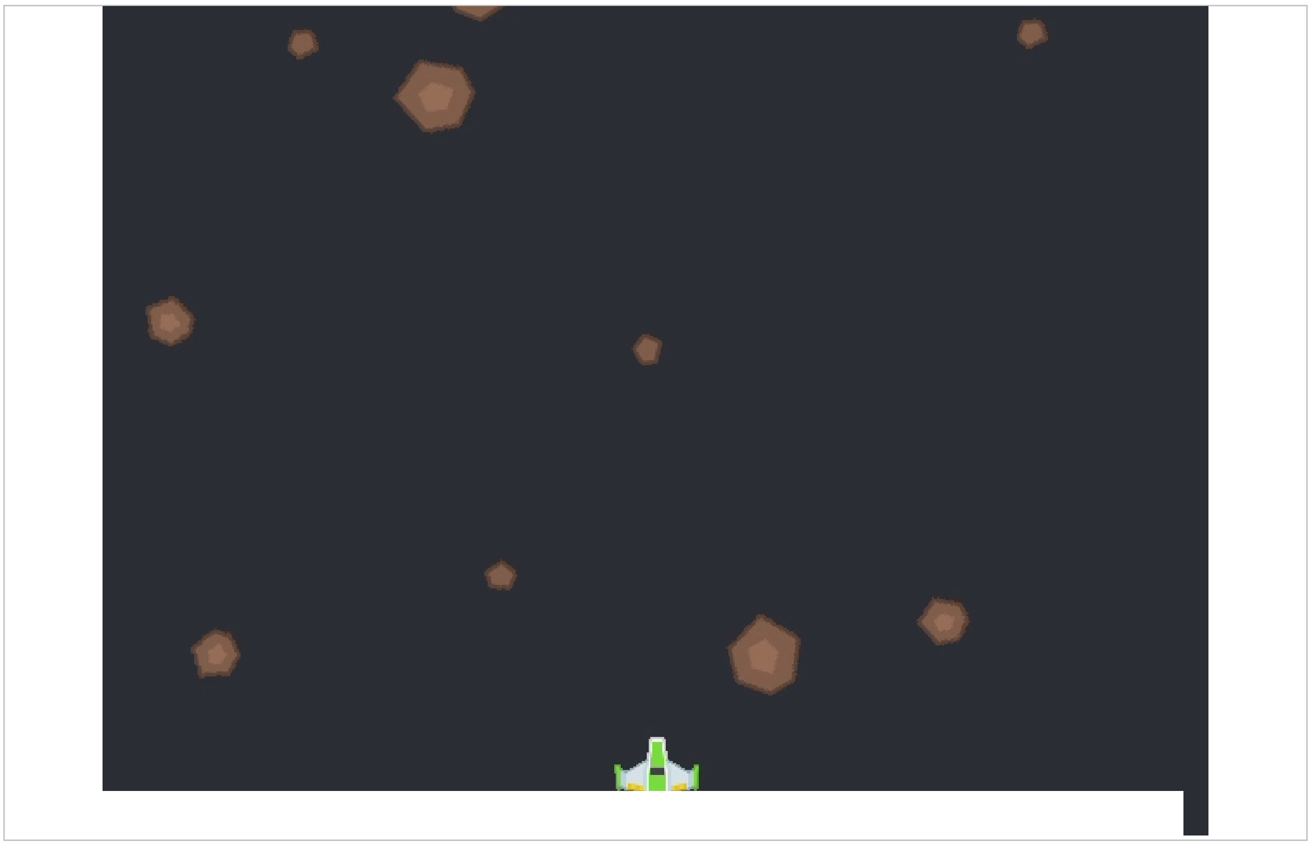
- notes = sprites-animating-random-images.pdf
- code = animatingsprites2.py

game example

- shooter0.7.py
- set random image for mob sprite object image
  - *random image from selection of image options*
  - *rotate and animate each random mob sprite image*

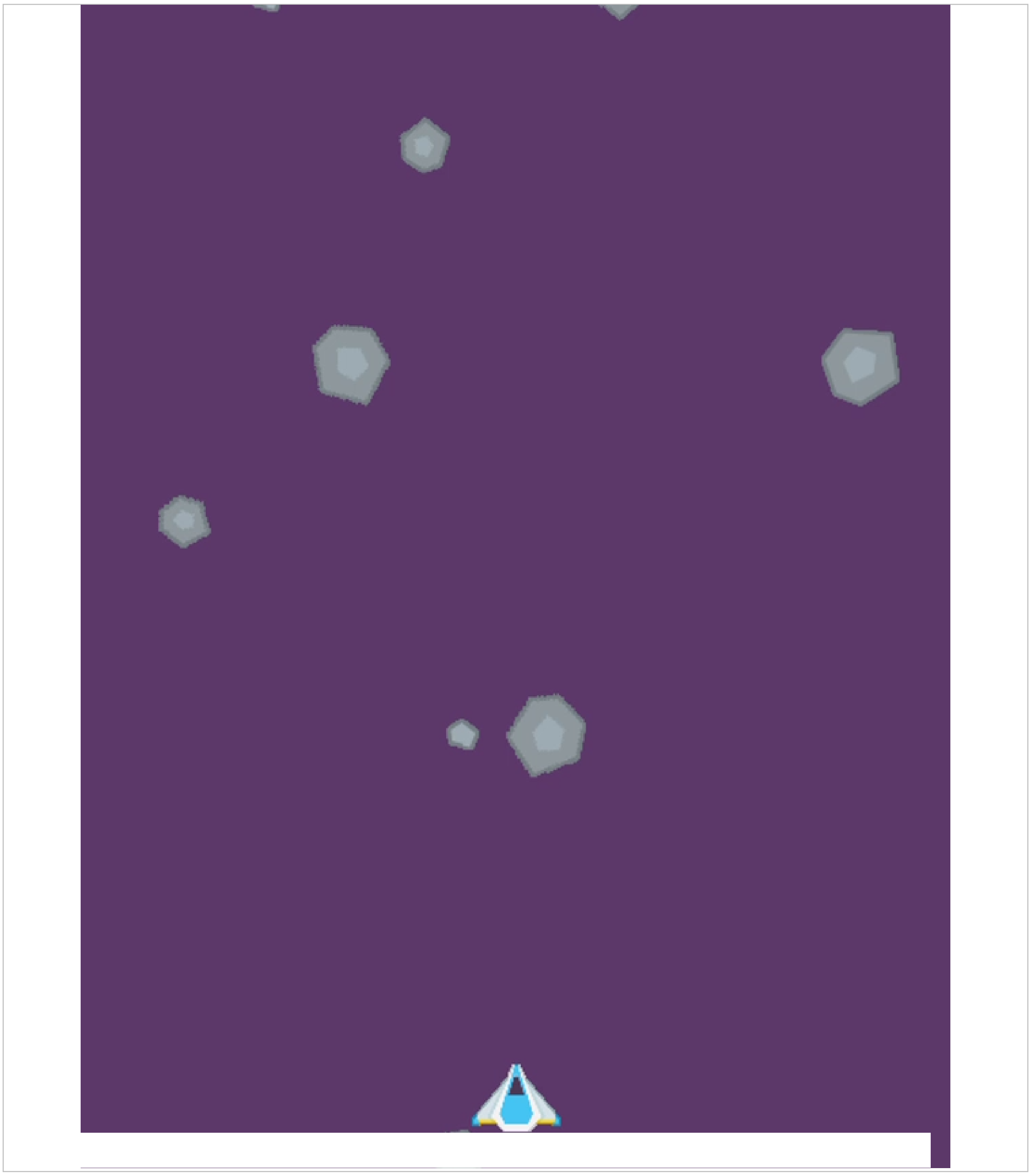
Video - Animating Sprites

random mob images



Video - Shooter 0.7

set random image for mob sprite object image



Resources

Demos

- pygame collision detection - better
  - *collisionsprites3.py*
- pygame animating sprites
  - *animatingsprites1.py*
- pygame random sprites
  - *animatingsprites2.py*
- pygame - Game 1 Example
  - *shooter0.5.py*
  - *shooter0.6.py*
  - *shooter0.7.py*

Games

- Animal Crossing
- Atari Asteroids
- Physics of a Bouncing Ball

Game notes

- Pygame
  - *sprites-collision-detection-better.pdf*
  - *sprites-animating-images.pdf*
  - *sprites-animating-random-images.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.
- various
- GameDev.net
- Video Game Design Schools
- gaming music covers
- Gaming music playlist 1
  - *Lindsey Stirling - Various Gaming Music Videos*
- Gaming music playlist 2
  - *Taylor Davis - Video Game Covers*
- covers include:
  - *Dragon Age, Halo, Zelda, Skyrim, Assassin's Creed, Mass Effect, The Witcher...*
- gaming inspirational music
- Really Slow Motion - YouTube Channel

Videos

- Algorithms - YouTube
- Animal Crossing
- Sid Meier's Civilization, Youtube
- The Last Startfighter, YouTube