Comp 388/488 - Introduction to Game Design and Development

Spring Semester 2017 - Week 5

Dr Nick Hayward

Contents

Python and Pygame - basic drawing

intro

- Pygame supports various options for drawing to a specified game window
  - *including defined functions for existing shapes*
  - *or custom shapes using lines and a mixture of shapes*
- built-in functions to help draw pre-defined shapes, e.g.
  - `rect` - *rectangle shape*
  - `circle` - *circle shapes drawn around a defined point*
  - `line` - *draw a straight line*
    - `aaline` - anti-aliased line
  - `lines` - *draw multiple contiguous lines*
    - `aalines` - anti-aliased lines
  - `polygon` - *draw a shape with a defined number of sides*
    - any number may be chosen...
  - `ellipse` - *a round shape contained within a rectangle*
  - `arc` - *draw a partial section of a standard ellipse*

Python and Pygame - basic drawing

drawing with `rect`

- we may start by adding a rectangle to our existing Pygame window, e.g.

```
pygame.draw.rect(window, WHITE, (200, 200, 100, 50))
pygame.draw.rect(window, CYAN, (300, 150, 100, 100))
pygame.draw.rect(window, MAGENTA, (400, 100, 100, 150))
```

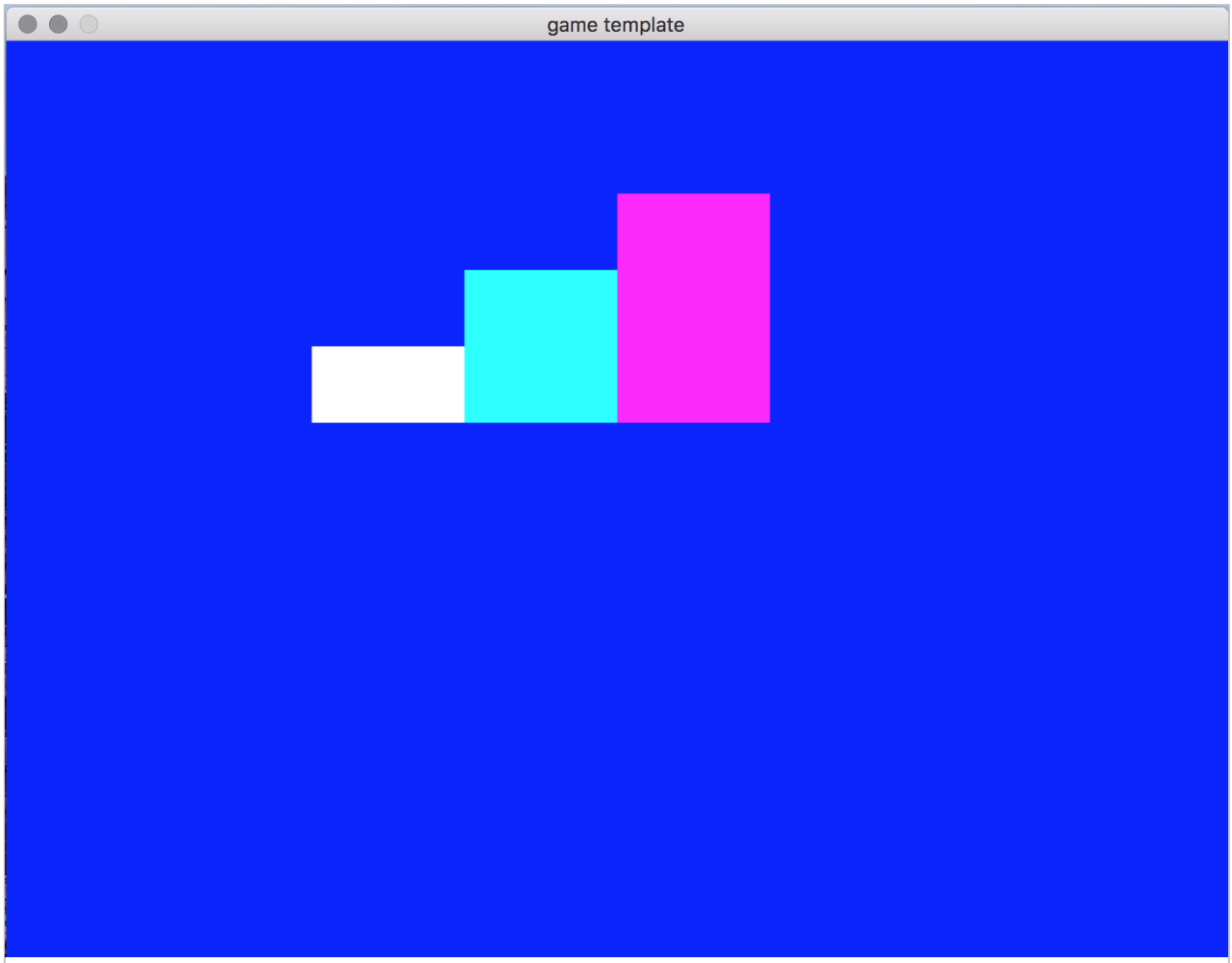
- parameters for these functions use the following example pattern
  - *where we want to draw the shape*
    - e.g. the Pygame window
  - *the RGB colour for our shape*
    - e.g. `(255, 255, 255)`
  - *pixel coordinates for drawing the shape*
    - x and y coordinates
    - x = position of left side of shape from left side of window
    - y = position of top of shape from top of window
  - *and the size of the shape*
    - width and height in pixels

```
pygame.draw.rect(WHERE, (R, G, B), (X, Y, WIDTH, HEIGHT))
```

- we may create some steps using the rectangles...

Image - Pygame Drawing

Example 1 - some steps



Pygame drawing - add some steps

Python and Pygame - colours

working with RGB - part 1

- clearly defined pattern we may use to work with colours in Pygame
- each colour is defined using the standard RGB primary colours
  - *Red, Green, Blue*
- we many then start to create our secondary colours
  - *as variants and combinations of these three primary colours*
- e.g.
  - *cyan = blue + green*
  - *magenta = blue + red*
  - *yellow = green + red*
- we may also create base colours for *black* and *white*. e.g.
  - *black = no colours*
  - *white = red + green + blue*

Python and Pygame - colours

working with RGB - part 2

- to create a particular colour
  - *defining how much of each primary colour we require mixing*
- mixing uses a known scale from **0** to **255** for each primary colour
  - *therefore giving a possible 256 points per colour on the scale*
- we may define some colours as follows
  - *red = rgb(255, 0, 0)*
  - *green = rgb(0, 255, 0)*
  - *blue = rgb(0, 0, 255)*
  - *cyan = rgb(0, 255, 255)*
  - *magenta = rgb(255, 0, 255)*
  - *yellow = rgb(255, 255, 0)*
  - *black = rgb(0, 0, 0)*
  - *white = rgb(255, 255, 255)*
- consider the sheer number of colour options for this scale
  - *256 * 256 * 256 =* ***16,777,216***
- over 16 million possible colour variations
  - *for our game's design and rendering*

Python and Pygame - basic drawing

drawing with `circle`

- also draw circles on a Pygame window using `circle()`
- instead of simply passing a width and height
  - *need to define a radius and a point around which the circle may be drawn*
- we can draw a circle, e.g.

```
pygame.draw.circle(window, WHITE, (150, 100), 30, 0)
```

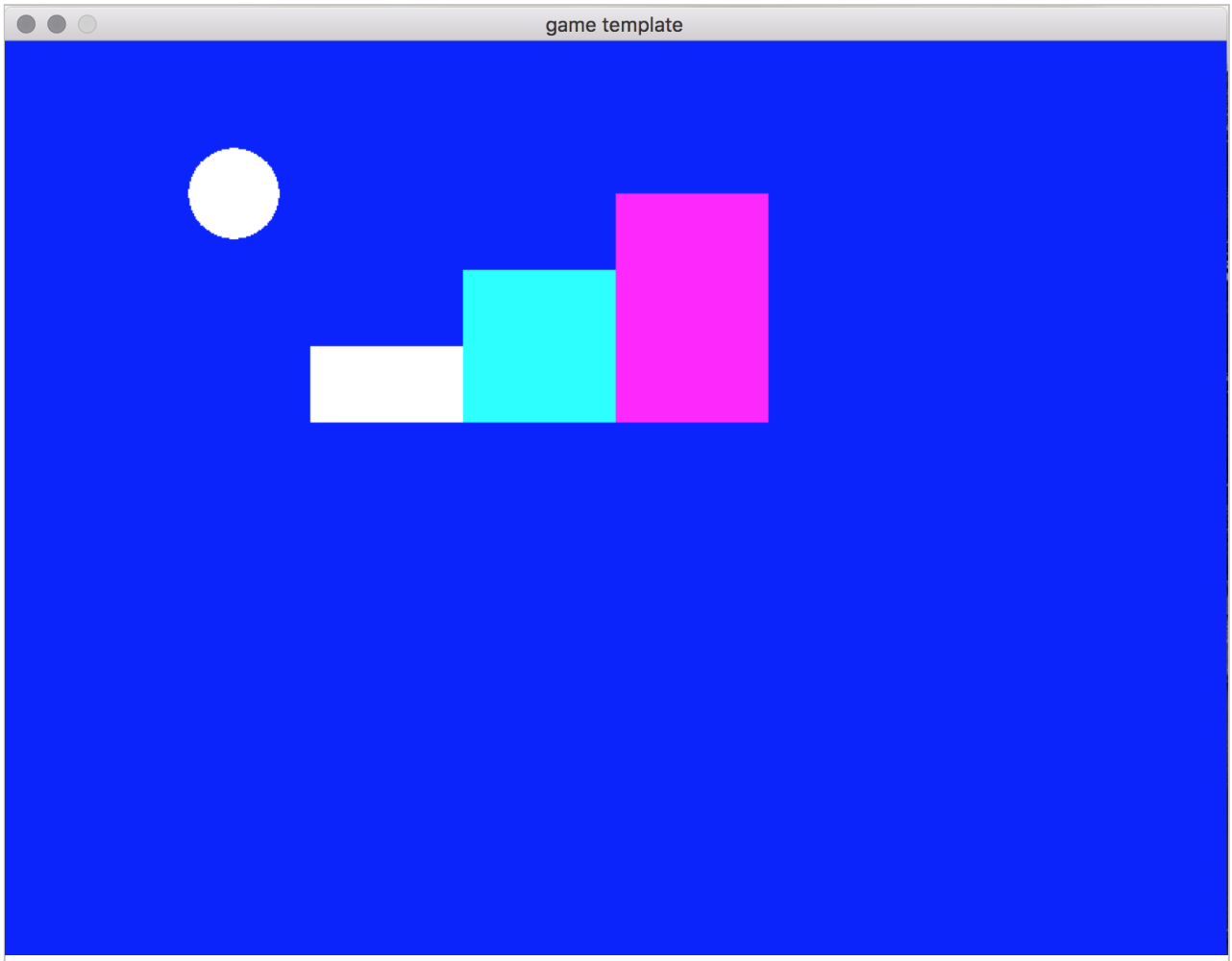
- this equates to the following parameters

```
pygame.draw.circle(WHERE, (R,G,B) (X, Y), RADIUS, LINE_WIDTH)
```

- `LINE_WIDTH` represents the width of the line used to draw the defined circle
  - *0 = a circle that is filled in with the defined RGB colour*
  - *2 = rendered circle would be empty with a 2 pixel wide outline*

Image - Pygame Drawing

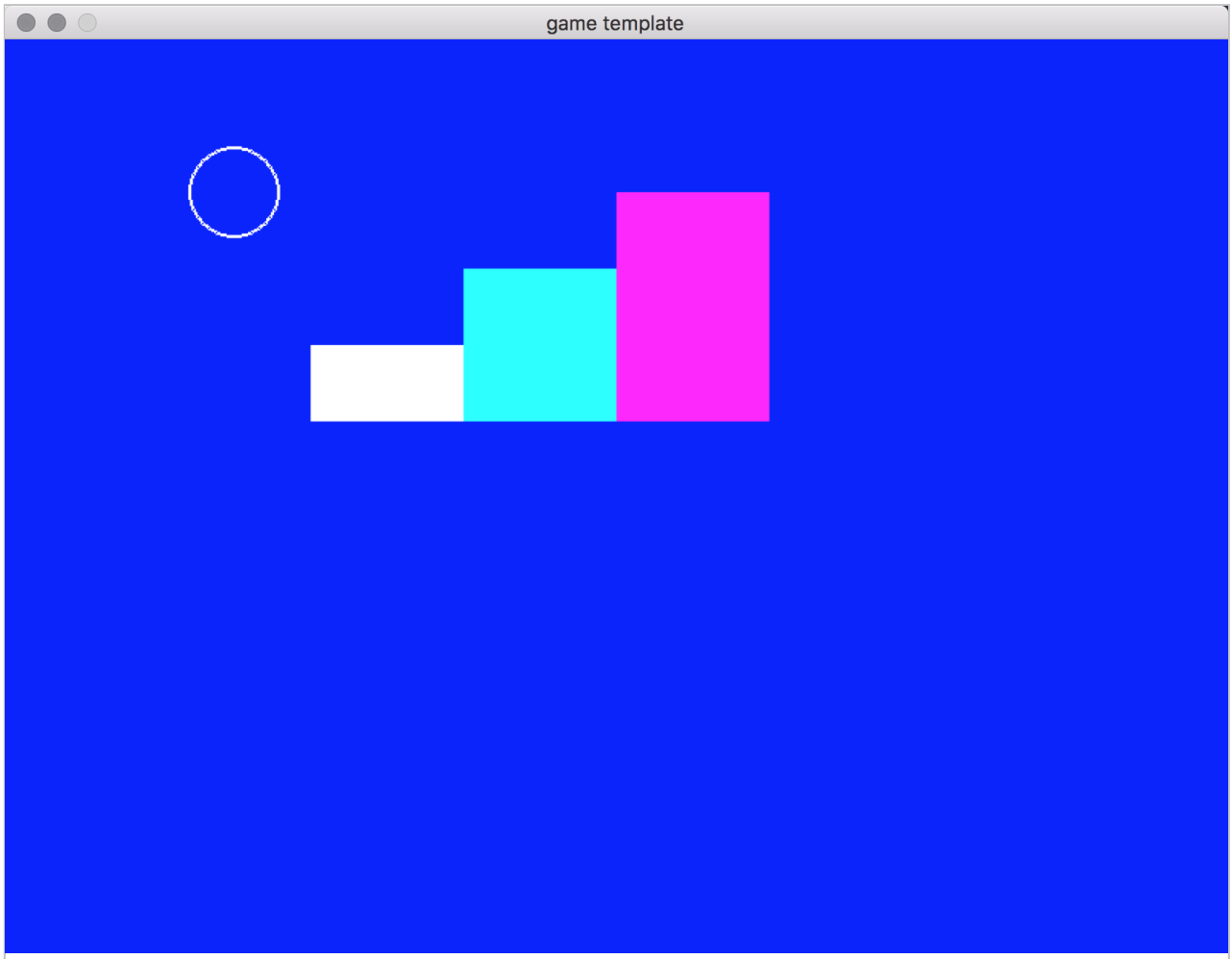
Example 2 - circle fill



Pygame drawing - circle with fill

Image - Pygame Drawing

Example 3 - circle outline



Pygame drawing - circle with outline

Python and Pygame - basic drawing

drawing with `ellipse`

- draw ellipses using a similar pattern to drawing a rectangle, e.g.

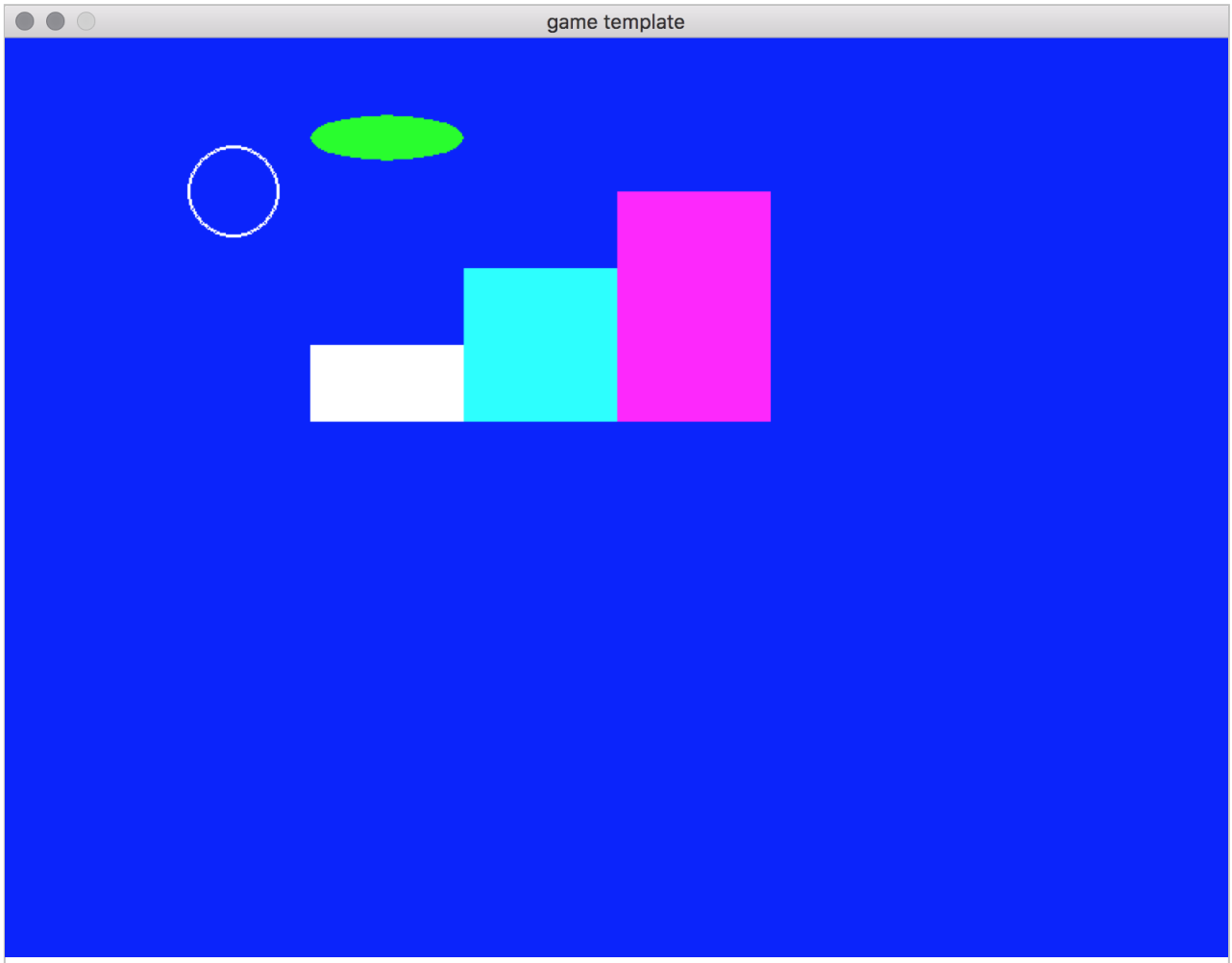
```
pygame.draw.ellipse(window, GREEN, (200, 50, 100, 30))
```

- also possible to create circles using an ellipse
  - *simply set the width and height parameters to the same value*
- create a rectangle with the same values as our ellipse
  - *the ellipse fits exactly within our rectangle, e.g.*

```
# draw an ellipse & containing rectangle - extra 2 is for width of rect
pygame.draw.ellipse(window, GREEN, (200, 50, 100, 30))
pygame.draw.rect(window, GREEN, (200, 50, 100, 30), 2)
```

Image - Pygame Drawing

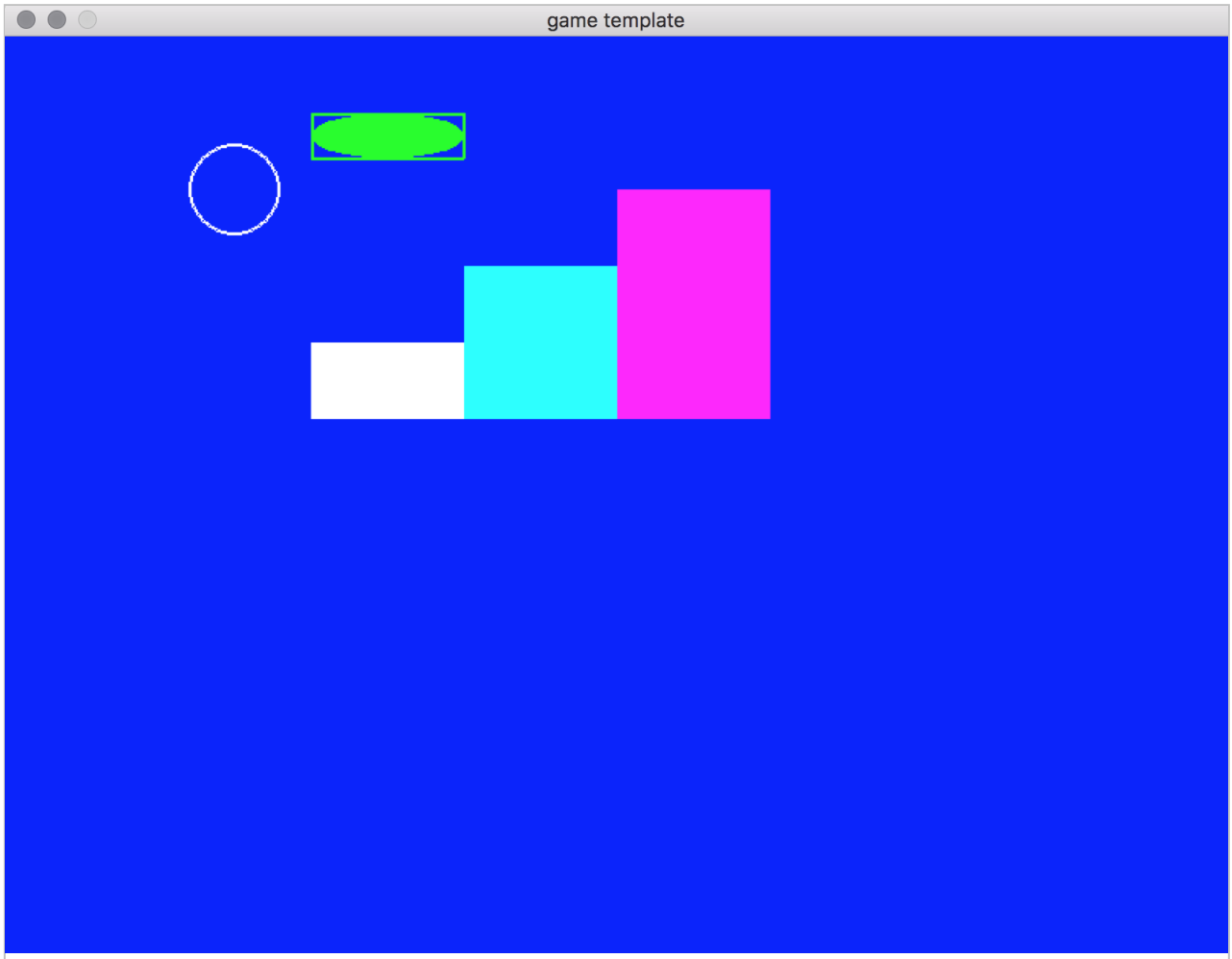
Example 4 - ellipse



Pygame drawing - ellipse with fill

Image - Pygame Drawing

Example 5 - ellipse with bounding



Pygame drawing - ellipse with bounding

Python and Pygame - basic drawing

drawing other shapes...

- create triangles and other shapes, including a pentagon, hexagon, or octagon
  - *can't use an existing function*
- instead, we need to use **paths**
  - *allow us to draw such irregular shapes*
- able to draw these shapes by defining points on our canvas
  - *then drawing lines between these points*
  - *filling the shape with a defined colour where necessary*
- draw a line using the following example,

```
pygame.draw.line(window, WHITE, (250, 250), (175, 175), 1)
```

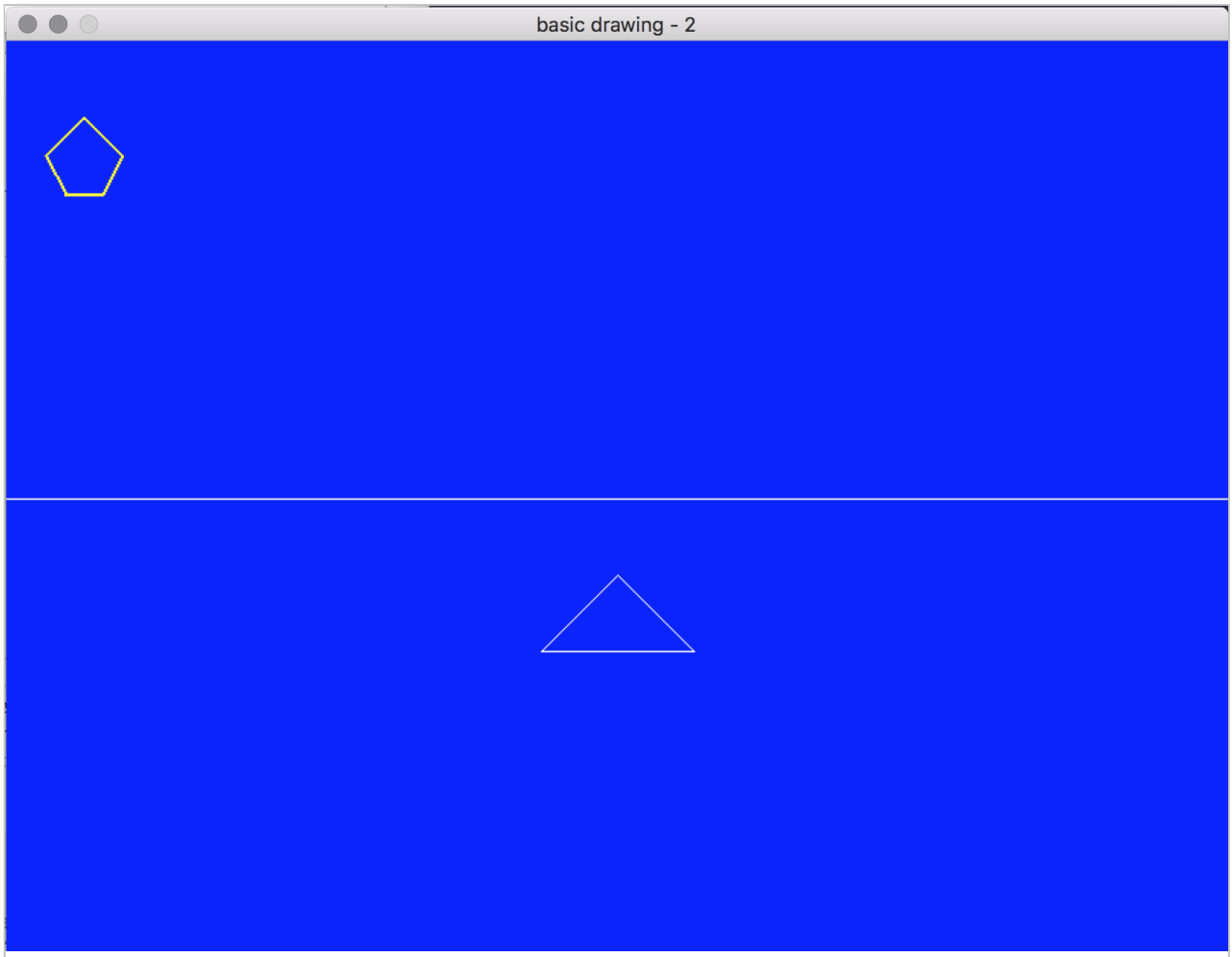
- same general pattern as other shapes
  - *add where to draw the line, and the RGB colour*
  - *add a tuple for the X and Y coordinates of the line's start position*
  - *add the X and Y coordinates for the end of the line*
  - *add the width of the line to draw*
- use `lines()` to combines multiple lines to draw various shapes, e.g.

```
pygame.draw.lines(window, WHITE, True, ((400, 350), (450, 400), (350, 40
```

- add `True` argument to tell Pygame to close the shape
  - *if set to `False`, the last line will not be drawn...*

Image - Pygame Drawing

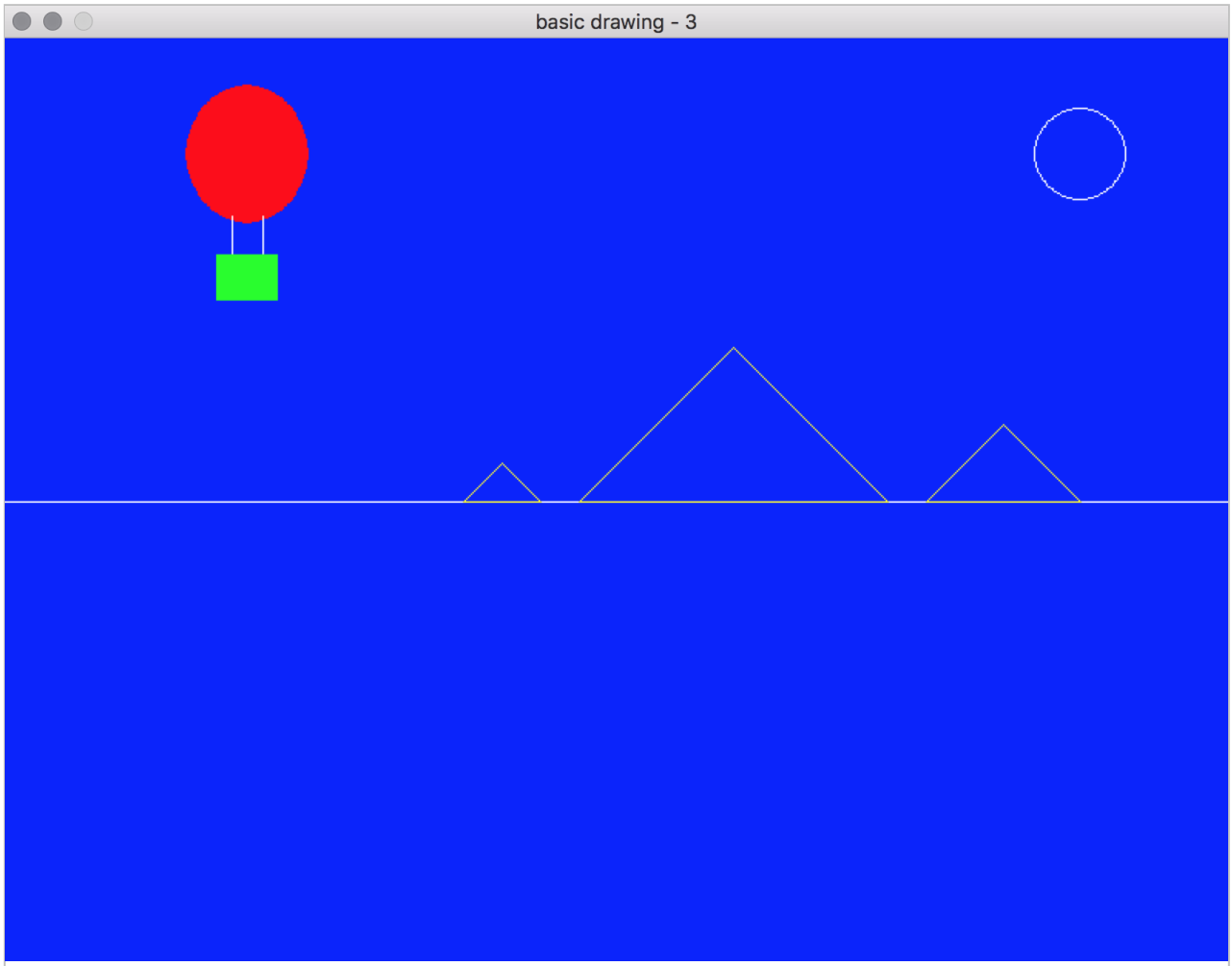
Example 6 - lines and shapes



Pygame drawing - line, triangle, pentagon

Image - Pygame Drawing

Example 7 - combo drawing



Pygame drawing - a mixture of shapes

Python and Pygame - moving shapes

basic animation - to the right

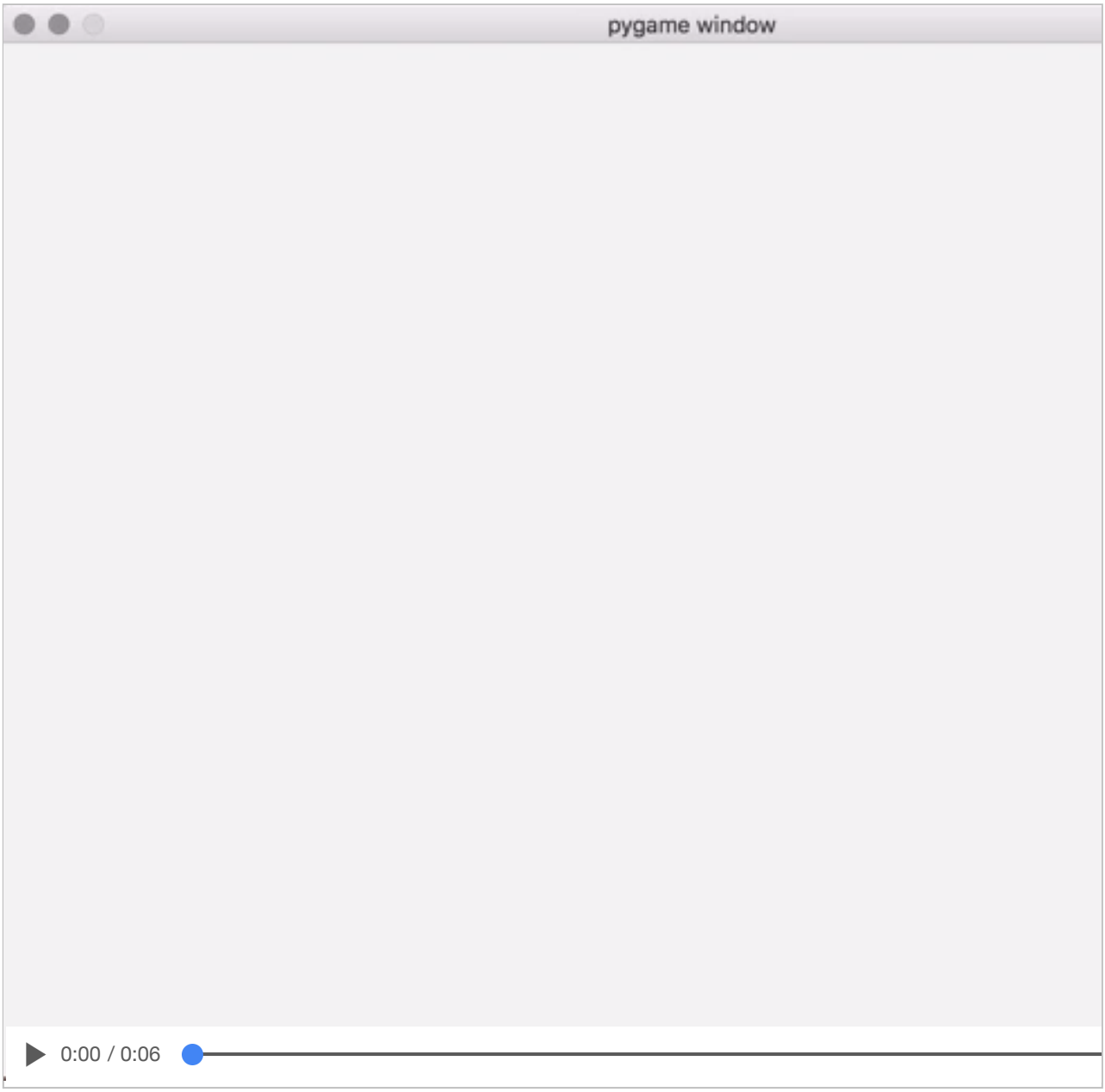
```
...
# rect coords...start at the centre
rectX = winWidth / 2
rectY = winHeight / 2

# create game loop
while True:
    # 'processing' inputs (events)
    ...
    # 'updating' the game
    # modify rectX by 4 pixels - higher creates impression of faster ani
    rectX += 4
    # 'rendering' to the window
    window.fill(WHITE)
    # draw rectangle
    pygame.draw.rect(window, GREEN, (rectX, rectY, 15, 10))
    # 'flip' display - always after drawing...
    pygame.display.flip()
```

- add some variables for the rectangle we want to animate
  - *set the X and Y coordinates to the centre of the window*
- then modify the *game loop*
  - *add 4 pixels to the X coordinate of the rectangle per update*
- then draw the rectangle to the game window as part of the rendering
- either update or flip the game window to show animation

Video - Moving Shapes

basic animation - move to the right



Python and Pygame - moving shapes

basic animation - different directions...

- make the rectangle move to the left side of the screen
  - *again, modify the value of the `rectX` variable*
  - *need to remove pixels to make it go to the left*

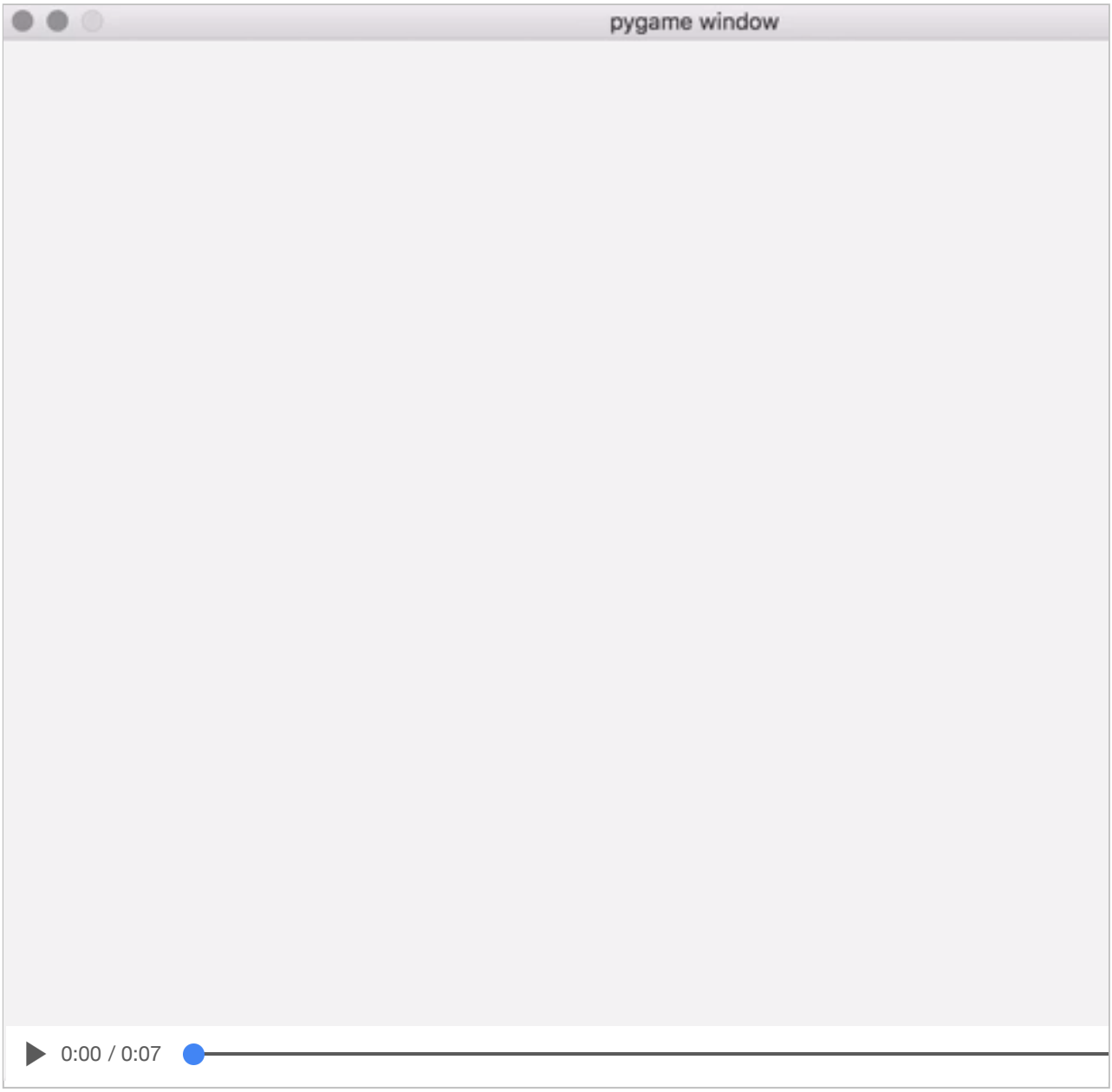
```
...
# modify rectX by 4 pixels
rectX -= 4
...
```

- also make our rectangle move at an angle
- might want to move it an angle down the screen
  - *add a variable for the vertical X and Y coordinates*
  - *incrementally modify to create the angle of animation down to the right*

```
...
# modify rect coordinates to create angle...to the right and down
rectX += rectVX
rectY += rectVY
rectVX += 0.2
rectVY += 0.2
...
```

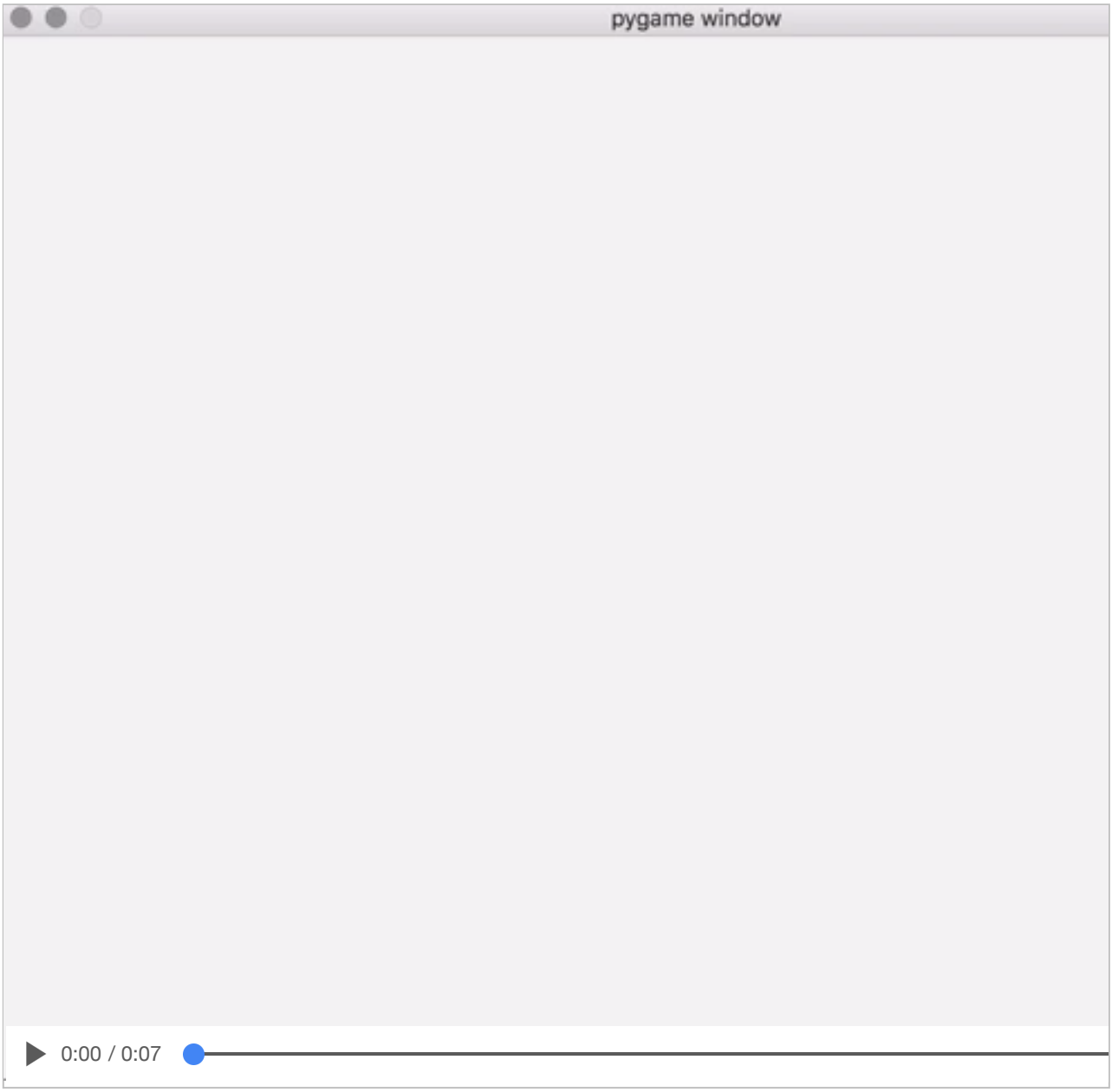
Video - Moving Shapes

basic animation - move to the left



Video - Moving Shapes

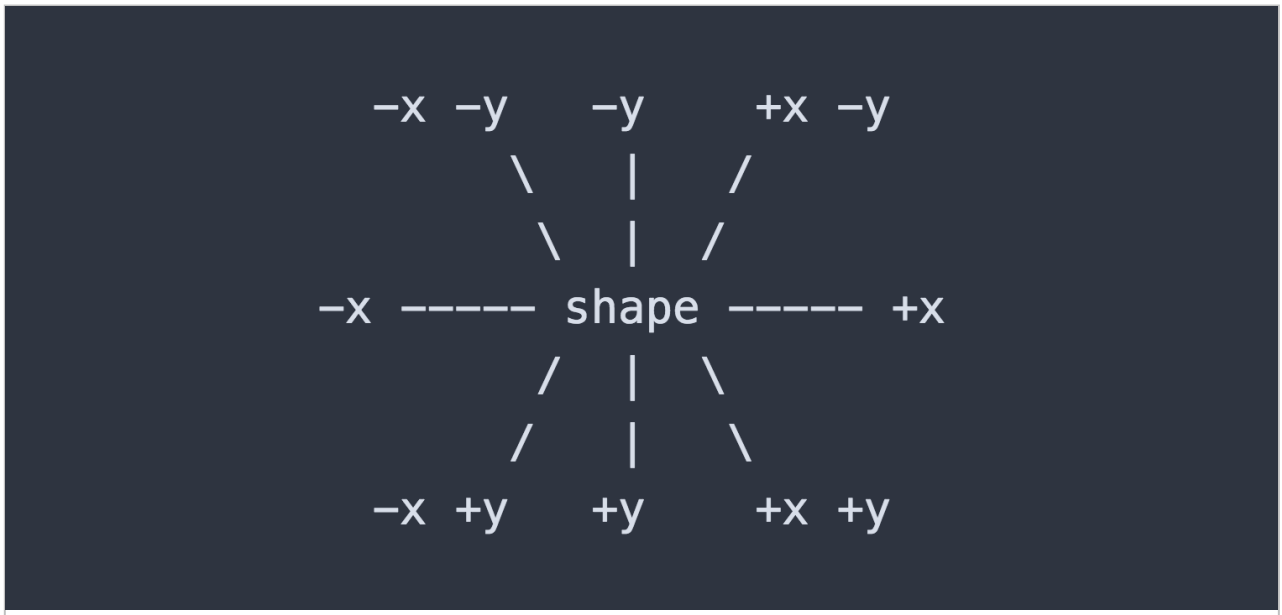
basic animation - move at an angle



Python and Pygame - moving shapes

basic animation - different directions...

- modify coordinates using the following pattern
- enough directions for our shapes to be able to recreate many classic games



Moving in different directions...

Python and Pygame - moving shapes

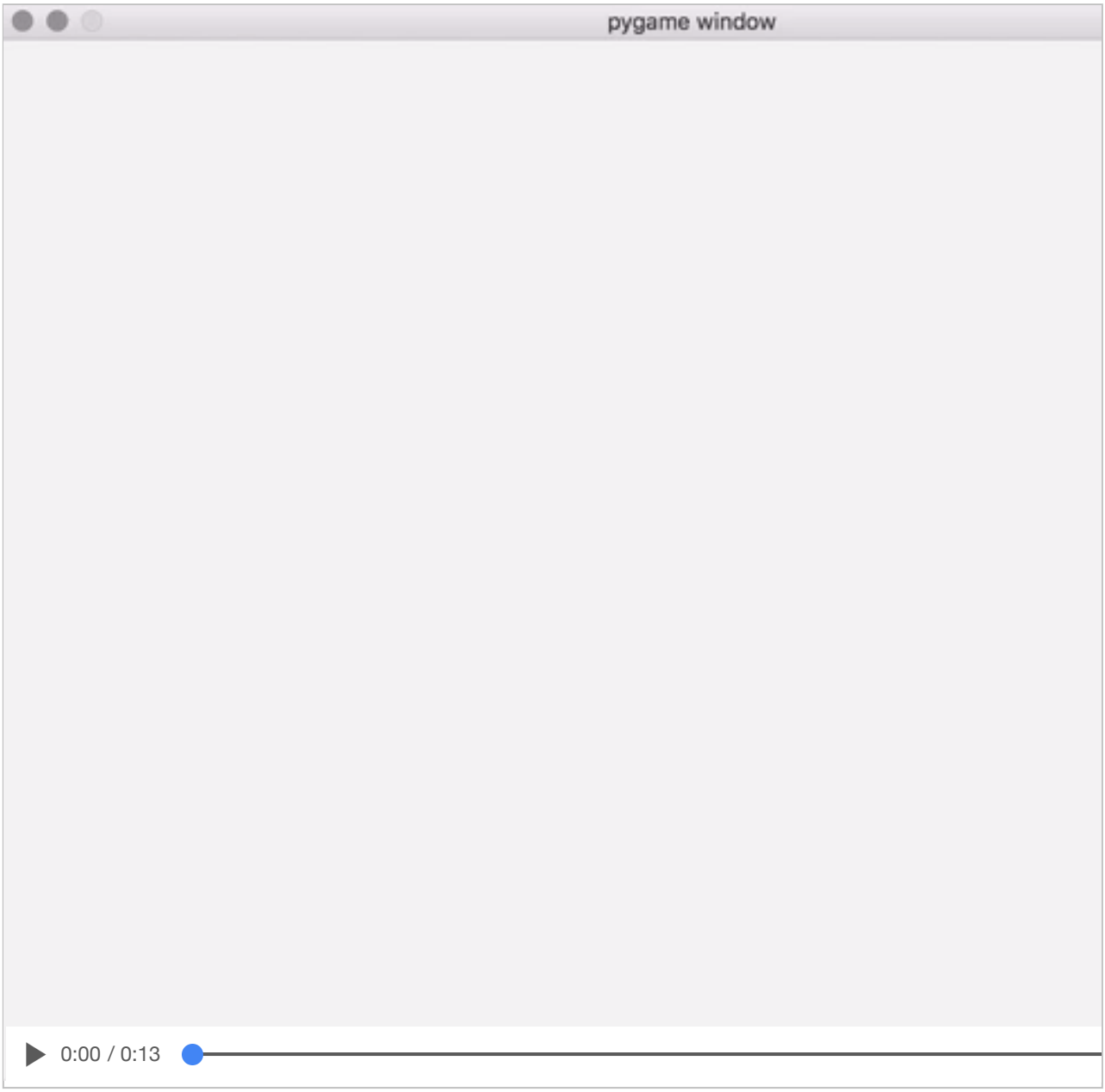
basic animation - check position

- as our shape moves across the screen
  - *may want to check that it doesn't simply disappear from one side*
- add a check for the position of the shape
  - *then reset its coordinates*
- e.g. if we animate our shape from the left side to the right side
  - *we may want it to keep moving from one side to the other*
  - *add a simple check for the value of the shape's X coordinate*
    - add to the *update* section of the *game loop*

```
# check position of rectX
  if rectX > winWidth:
      rectX = 0.0
```

Video - Moving Shapes

basic animation - move from side to side



Python and Pygame

extras

- more Python and Pygame next week
  - *more drawing, animation, interaction and controllers...*
  - *game demo*
- check course GitHub account for
  - *extra notes, various documents - notes*
  - *examples & templates - source*

Games and formal structure

procedures

- player's consideration and perspective of gameplay and objectives
  - *predicated on a clear understanding of procedures and rules*
- for example,
  - *to be able to act as the player in the chosen game*
  - *to actually know what they can and can't do to complete defined objectives*
- procedures allow us as designers and developers to clearly define
  - *how the player may interact with the game*
  - *and modify the interactive nature of the game*
- e.g. in *Draughts*, each player is allowed to
  - *pick up their own pieces*
  - *then physically move them around the board*
  - *they may also stack pieces*
  - *remove their opponents pieces...*
- e.g. in *Space Invaders*, each player may interact with a physical device
  - *to control their spaceship*
  - *fire their cannon*
  - *select game options...*
- such procedures may be abstracted from the game specific rules

Games and formal structure

rules

- a game's rules may be simple or complex
  - *sometimes to the point of a short novel*
  - *but their intention still remains the same*
- creating a set of clearly defined parameters
  - *what a player can and cannot do to achieve the game's objectives*
- rules may also be used to clarify
  - *what does and does not happen when patterns are matched in a game*
- e.g. in *Draughts*, by completing a certain move
- e.g. in *Space Invaders*, by successfully killing all of the advancing aliens
- some of these rules may be used to define objects
  - *such as the pieces in Draughts or the weapons in Space Invaders*
- others may deal with gameplay concepts
- the very nature of procedures and rules infers a sense of authority
  - *they still require additional structures to enforce them within the game*

Games and formal structure

boundaries

- boundaries help us enforce certain procedures and rules
- using boundaries, to some extent, we may ensure that players of our game
  - *need to adhere to rules to be able complete their objectives*
- e.g. in *Space Invaders*, such boundaries may be physical or digital
  - *restricting the player to a given interaction option*
  - *or certain scope or movement in a game's level*
- such boundaries are creating the imaginary realm of the game
  - *where the rules apply to affect the game's objectives.*
- boundaries help us create the immersive nature of the game
- consider VR and AR
  - *we start to see how new boundaries modify our perceptions*
  - *perceptions of procedures, rules, and gameplay itself*

Games and formal structure

conflict, challenge, battle...

- conflict will often be an active part of playing a game
  - *due to certain objectives within our game*
  - *an indirect consequence of rules we define for the game*
- may also occur in both single player and multi-player games
  - *it will necessarily manifest in different ways*
- we may create such conflict using defined structures of the game
  - *challenging the player with the underlying procedures and rules*
- as a player masters a given part of the game
  - *the conflict will then start to diminish*
  - *or simply be replaced by another problem or situation to resolve*
- e.g. in *Draughts*, initially faced with a direct conflict between players
  - *by simply moving and positioning pieces one player against another*
  - *then, one player starts taking another player's pieces...*
- rules of the game have created the potential for conflict
  - *each player directly challenges the other by leveraging available rules*
- such conflict is another useful tool for modifying gameplay
  - *then modifying difficulty and challenges as a player progresses through a game*
- objectives of a player often conflict with the rules and procedures
  - *may often intentionally limit and guide behaviour within a game*

- by resolving such conflict
  - *a player is able to achieve their desired objectives*
  - *hopefully, the game's overall object as well*

Games and formal structure

outcome, end result...

- another noticeable similarity between games
  - *the simple opportunity for an outcome*
- may include a defined winner, a loser, a draw...
  - *even the simple fixed ending of a story, saga or quest*
- some games may represent such an outcome and end result as either
  - *stay alive and win or die and lose*
- such outcomes may often be a natural conclusion to the defined rules
  - *and the primary, over-arching objective of the game itself*
- however, it doesn't always need to be so clear cut
  - *the end of one adventure, but the beginning of another*
  - *Tolkien-esque in scope and consideration*
- also clear distinction between a game's various objectives and defined outcome
- e.g. in *Space Invaders*, we may see many objectives for a player
  - *destroying aliens, maintaining lives, advancing through different levels...*
- in *Space Invaders*, the single outcome is to
  - *successfully complete each level to complete the game*
- how we use such objectives towards the overall outcome
  - *is an option we can use to modify gameplay itself*
  - *and the overall experience of our game*

- in multi-player games, a key component of a game's outcome
  - *includes the palpable sense of uncertainty*
- as we increase conflict and competition
  - *uncertainty will likewise be increased*
  - *becomes a key factor in encouraging player's to return to a game*

Image - Create a memorable ending

Super Mario Bros. vs Castlevania

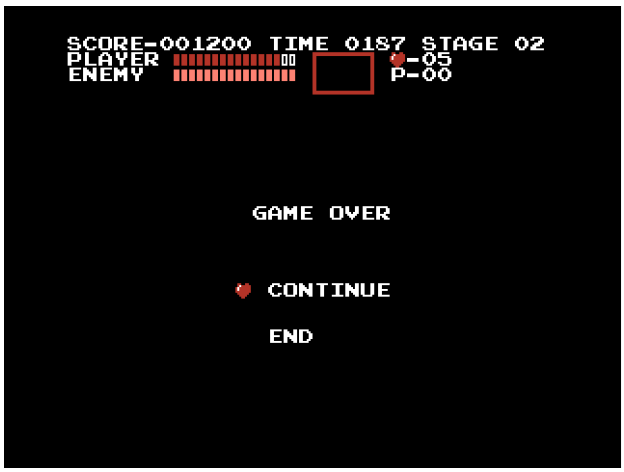
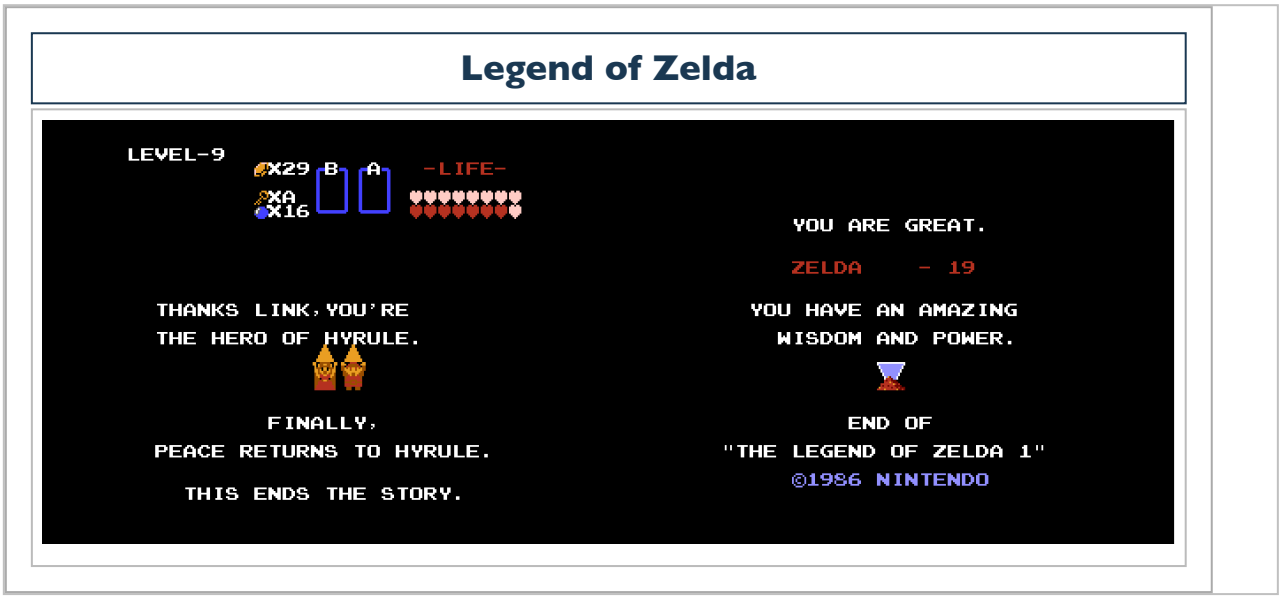
Super Mario Bros.	Castlevania
	

Image - Create a memorable ending

Legend of Zelda



Game example - Space Invaders

a classic bit of fun...

- Space Invaders - Sega and Taito
  - *close fidelity example from 1985 - graphics almost identical to original 1979 version released in Japan*
  - *streaming version of game*
- Draughts/Checkers
  - *playable version*

Games and engagement - learning to play again

concept & premise

- formal structures for a game may also benefit from a concept or premise
  - *helps frame or wrap the general gameplay*
- we're creating an underlying reason for a given game
- something we hope our players will enjoy
  - *and consider worthy of their time and investment*
- a concept or premise is a great way to hook our players into a game
- player needs a valid reason to play the game
  - *rarely just the mechanics...*
- **Space Invaders** has a simple hook for the game and play
- for **Mario** games, Miyamoto uses a simple premise to wrap the mechanics and gameplay
  - *progress through varied levels to save the Princess*
  - *a means of showcasing each game's formal structures*

Games and engagement - learning to play again

story and characters

- development of video games includes a shift in design and story telling
  - *e.g. towards a consideration of characters*
- character development has been growing since the earliest games
  - *a useful, fun part of developing a premise for a game*
  - *Mario, Donkey Kong, Sonic, Pac Man...*
- each character development acted as a tool to help engage players
- characters help a player to become engaged and immersed
  - *in the general premise of a game*
  - *a specific story in particular*
- characters act as a direct link and interaction
  - *between a game's narrative and its player*
- designers and developers may also use characters
  - *a means to manipulate stories, and general gameplay...*
- a player may impart their own characteristics and personality on a game character
  - *need to be careful not to restrict a character too much*
- players often deeply invested in a game due to characters
  - *they form an attachment with characters...*
  - *conventions and cosplay continue to grow in popularity*

- story and characters may fulfill a dramatic context within our game
  - *depends how far we wish to push such elements within our game*

Games and engagement - learning to play again

pushing boundaries

- many examples we may reference as archetypal games
- many games break this mould
  - *may even push the standard perception of a game*
- recent development in games is towards the use of immersive environments
  - *simply promote calm and relaxation.*
- gaming to reduce stress by exploration
  - *instead of high paced action and adventure*
- a natural progression from earlier games, e.g. *Civilization*, *Age of Empires*...
  - *to a new audience and emerging genres*
  - *Abzu, Journey, Proteus...*
- annual *Games for Change* festival in New York
  - *considers games in a broader social context*
  - *Games for Change*
- boundaries are also being pushed with indie development and experimental gaming concepts
- *Independent Games Festival*
  - *a great place to start exploring such ideas and concepts*
  - *Independent Games Festival*
- *IndieCade* festival, the International Festival of Independent Games

- *IndieCade*

Video - Abzu

Source - Abzu trailer - YouTube

Games and development

Enter the Mummy's Tomb - objects, attributes...

- in our earlier game, *Enter the Mummy's Tomb*, we introduced three initial characters
  - *explorer (our Egyptologist)*
  - *high priest*
  - *scary pharaoh*
  - *the mummy*
- objects may include known characteristics and attributes from real world, e.g.
  - *name*
  - *health*
  - *current value & status, lives, regeneration...*
  - *physical characteristics*
  - *height, speed, strength, vision...*
  - *skills*
  - *fighting, shooting, intelligence (problem solving &c.) ...*
  - *motion*
  - *e.g. walking, running...*
  - *actions*
  - *pick-up, throw, move, drop...*
- each character possesses such attributes, to a greater or lesser extent...
- may also reuse such attributes as a definition, template
  - *help guide the subsequent development of other characters*
- new characters might include
  - *earth-bound creatures*

    - horse, scarab beetle, snake...
- *Egyptian gods*
- *e.g. Anubis, Osiris, Isis, Horus, Ma'at, Sekhmet, Seth...*
- *enemies*
- *allies...*
- *other explorers...*

Games and development

Enter the Mummy's Tomb - attributes...

- consider attributes useful and applicable to each of our main characters
  - *characteristics and actions our characters may need and use in the game*

explorer	high priest	scary pharaoh/mummy
name	name	name
health	health	health
fight/attack		fight/attack
	help/aid	
info	info	info
retreat		retreat

- list of attributes is not exhaustive, and it may grow as we develop a game
- may also find it useful to combine some of these attributes into a given class
  - *fight and health attributes may only apply for an **attack** method*
- may also consider the tombs as an additional object within our game
  - *attributes may include, e.g.*

tomb
name/number (e.g. KV17)
owner

tomb
owner type
find treasure
info

- may start to see common attributes and characteristics
- create methods to help us structure and call such characteristics within a given class
  - *e.g. a class for the explorer*
- *owner* of each tomb is unknown until we randomly pick a character
  - *may be an instance of the high priest or the mummy class*
- *owner type* may end up either helping or attacking the *explorer*

Games and development

Enter the Mummy's Tomb - initial structure

- many of these objects share common traits and attributes
  - *explorer, high priest, and mummy may use inheritance*
- allows us to create a useful superclass/parent class
  - *this will be our initial* ***GameCore***
- *GameCore* may include the following:
  - *attributes*
    - name
    - owner
    - health
  - *methods*
    - fight/attack
    - help/aid
    - info
    - retreat
- add to the *GameCore* as we build out our current game
  - *each of the characters may inherit from this GameCore class*
  - *each character class may also override default methods*
- for example
  - *give the explorer enhanced options to fight/attack*
  - *perhaps the mummy will have a higher initial health value*
- a *tomb* may also inherit certain default attributes from the *GameCore*
  - *including name and info*
- each *tomb* will also contain, or be composed of, another object

- *such objects may be used to perform specific tasks*
- *perhaps an owner composed of a high priest or mummy*

Games and development

quick exercise

consider the following 4 characters:

- poet / bard
- archer
- scout
- knight

then outline the following:

- abstract objects and attributes for all of these characters
- show developer pattern from abstract to specific character
- show relationship between character objects and attributes
- similarities and differences between developer and player updates
  - *for abstract and specific characters...*

Games

- Abzu
- Journey

References

- Draughts
- Space Invaders

References - Pygame

- pygame.event
- pygame.locals

Videos

- Abzu trailer - YouTube