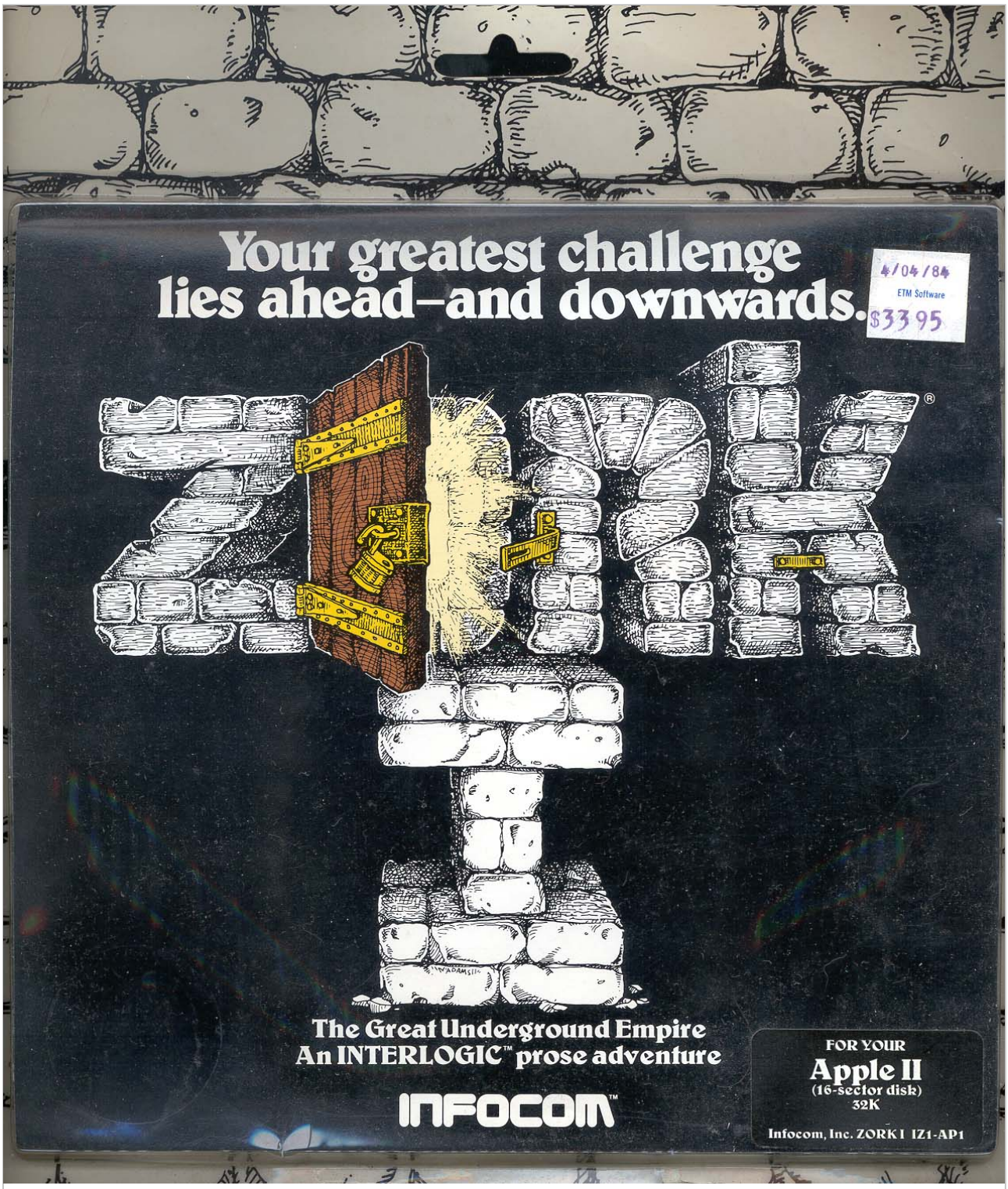
Comp 388/488 - Game Design and Development

Spring Semester 2019 - Week 4

Dr Nick Hayward

Image - Zork



Zork

Games and planning

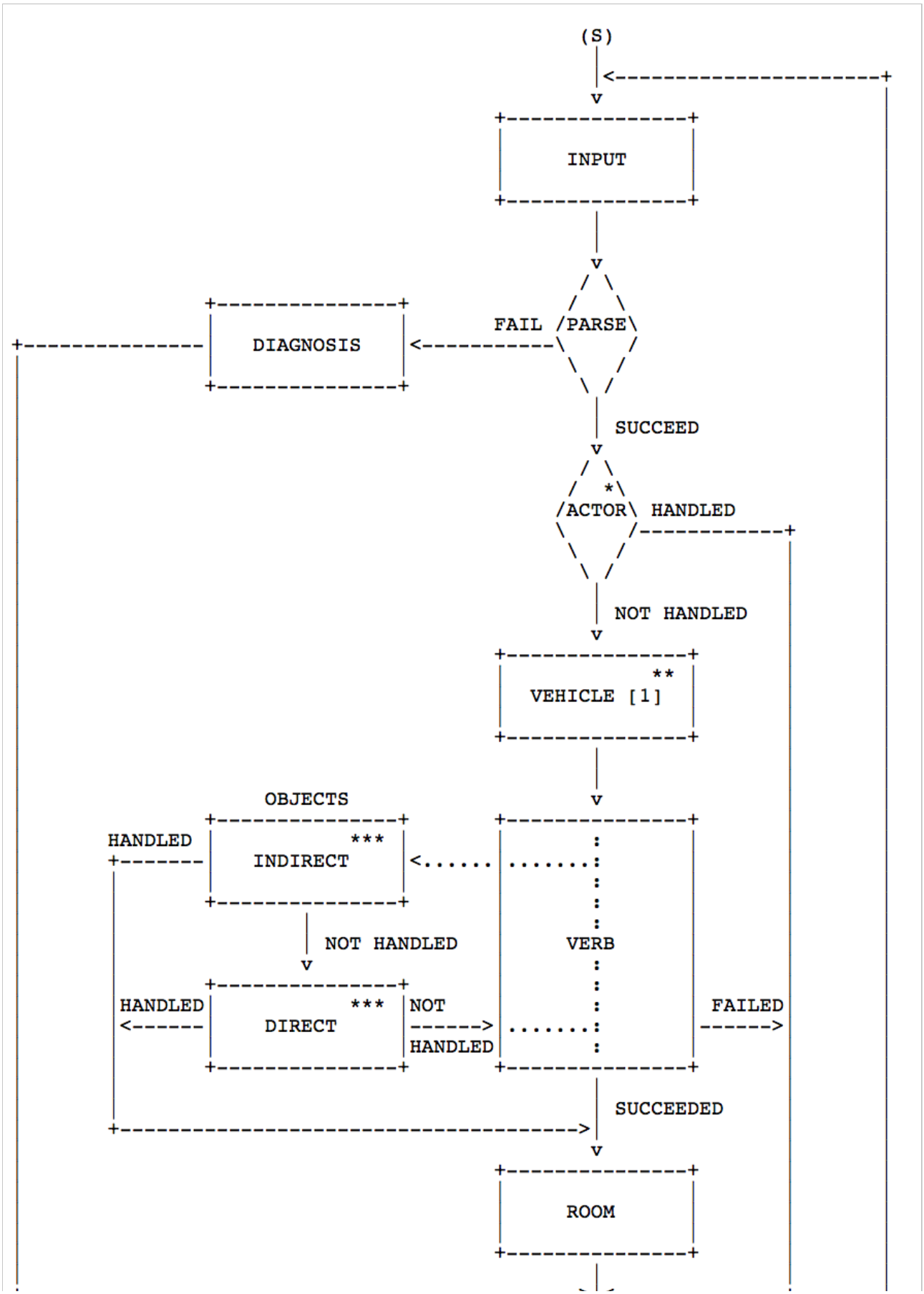
Zork

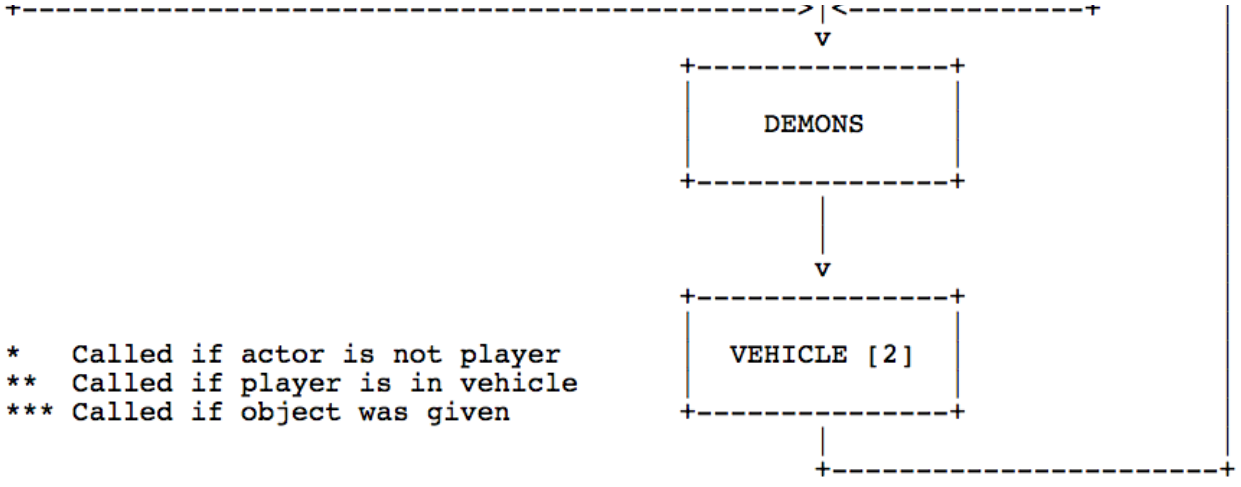
- **Zork**, one of the best known text-based adventure games
  - *written in 1977 for the PDP-10 mainframe computer*
  - *second text-based adventure game ever written - first was Colossal Cave Adventure*
    - written in 1976 for the PDP-10
  - *both games were interactive fiction*
  - *set in the ruins of an ancient empire lying far underground*
- player's character is simply an anonymous adventurer
  - *who is venturing into this dangerous land in search of wealth and adventure*
- primary goal of this game is to return alive
  - *from exploring the "Great Underground Empire"*
- a victorious player will earn the title of *Dungeon Master*
- game's dungeons include a variety of objects...
  - *interesting and unusual creatures, objects, and locations*
- best known creature is the ferocious but light-fearing *grue*
  - *a term for a fictional predatory monster that dwells in the dark*
- ultimate goal of Zork I is to collect the Twenty Treasures of Zork
  - *and install them in the trophy case*
- finding the treasures requires solving a variety of puzzles
  - *such as the navigation of two complex mazes*
- end of Zork I becomes the entrance, and beginning to the world of Zork II
- fantastic text-based game
  - *feels part fantasy, part classical mythology, and part sci-fi...*
- Download the Zork games for Mac and Dos/Windows at the following URL,

- *Infocom - Zork*

Image - Flowchart - Example 2

Zork

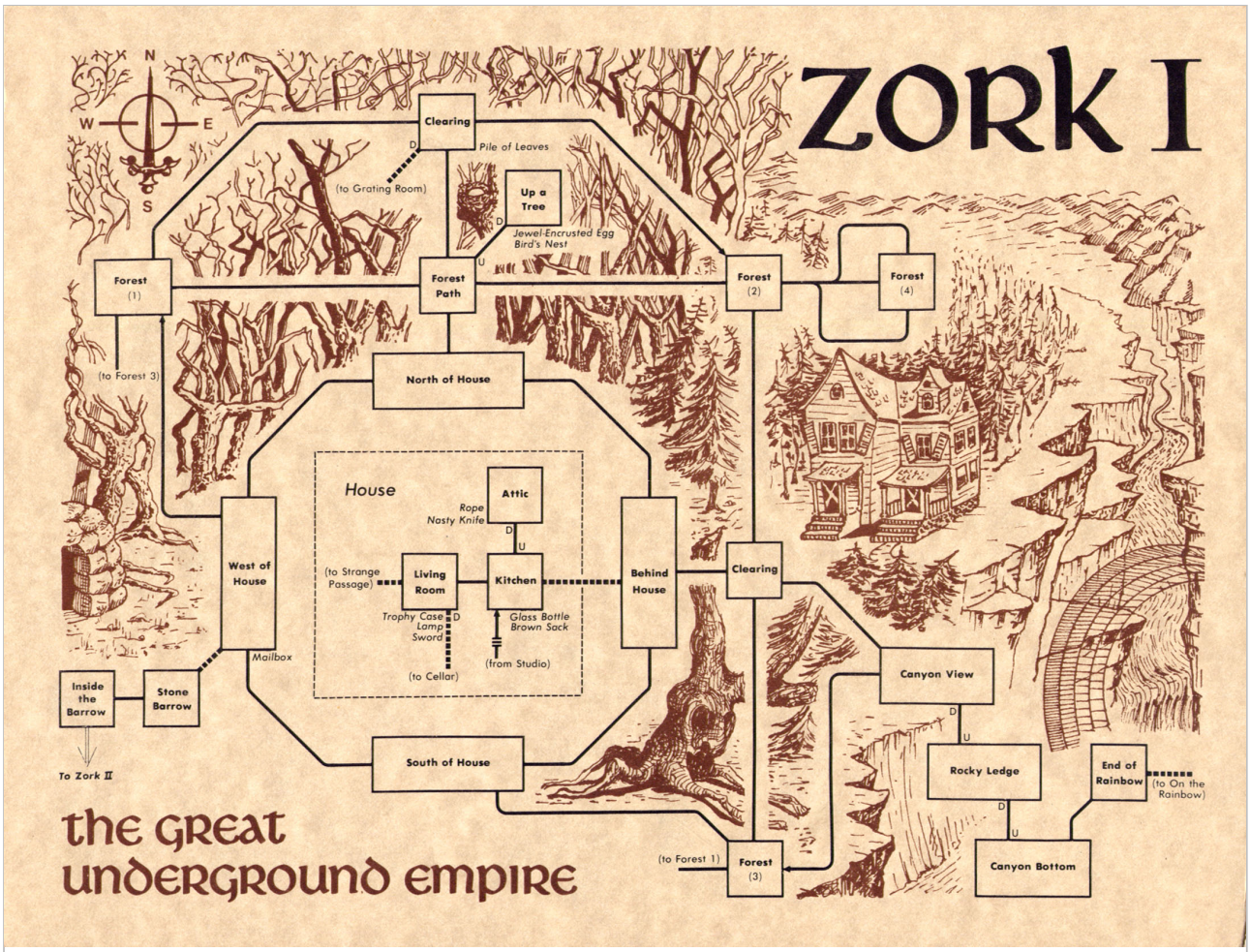




Flowchart - Zork - Logic

Image - Flowchart - Example 3

Zork Map



Flowchart - Zork - Map

Games and planning

A bit of fun - Zork

- MS-DOS - https://archive.org/details/msdos_Zork_I_-_The_Great_Underground_Empire_1980
- Playstation example - https://archive.org/details/psx_zork
  - *released in 1996 in Japan*

One leader, and three advisors...

Python and Pygame

intro

- a brief consideration of development, specifically with Python and Pygame
- install instructions for Python 3.x and Pygame
  - *Python & Pygame setup - OS X*
  - *Python & Pygame setup - Windows 10*
- Pygame is a powerful and useful set of modules to help develop and create games with Python
- best place to start is simply by visiting the website for the Pygame modules
  - *Pygame - Getting Started*

Python and Pygame - game development

game template - intro

- we may create a template file for starting our Pygame based projects
  - *a number of ways to setup a template for such game based development*
- there are a few common requirements we may start to add to a template, e.g.
  - *import required modules*
    - e.g. *pygame*, *sys*...
  - *define default settings for a Pygame window*
  - *initialise Pygame*
  - *setup the required game loop*
    - logic executed for each frame in our game...
  - *process inputs*
    - listen for events within the game
    - track events with Pygame
  - *update the game for any required changes...*
  - *rendering of the game and its graphics*
    - draw the game to the Pygame window
  - *monitor frames per second (FPS)*
    - optional for template
    - where applicable for a given game...

Python and Pygame - game template

part 1 - import

- start by importing Python modules
  - *e.g. pygame module*

```
# import modules for pygame template
import pygame
```

- we may also import the `sys` module
  - *may use as a way to exit the game*

```
# import modules for pygame template
import sys
import pygame
```

Python and Pygame - game template

part 2 - window defaults

- then add some defaults for a *window* in Pygame
  - *defining our variables as follows*

```
# variables for pygame
winWidth = 800
winHeight = 600
FPS = 30
```

- we're setting the default window size - width and height
  - *and the frames per second for the game*
- FPS may be added for applicable game types
  - *sets how fast the game will update per second on each system*
  - *may update this value for each game's requirements*
- **game loop** will then reflect the number of frames per second
  - *loop will now run 30 times per second*
  - *e.g. each loop is set to 1/30 of a second*

Python and Pygame - game template

part 3 - initialise

- then add general initialisation for our game's initial settings
  - *start by initialising Pygame, and the sound mixer*
  - *sound mixer allows us to play back sound at various points in our game*
- then create our screen or window for the game
  - *and add a brief caption for this window*
- if we're going to define the FPS for our game
  - *we also need to define a clock*
- clock helps us track how fast the game is going
  - *allows us to ensure that we're maintaining the correct FPS*

```
# initialise pygame settings and create game window
pygame.init()
pygame.mixer.init()
window = pygame.display.set_mode((winWidth, winHeight))
pygame.display.set_caption("game template")
clock = pygame.time.clock()
```

Python and Pygame - game template

part 4 - game loop

- now setup and initialised the basics for our template
  - *need to add a basic game loop to our Pygame template*
- **game loop** is one of the key requirements for developing a game
  - *including with Python and Pygame*
- *Game loop* is executed for every frame of the game
  - *three processes will happen as part of this loop*
- **processing inputs** (aka events)
  - *responding to interaction from the player with the game*
  - *e.g. keyboard press, mouse, game controller...*
  - *listening for these events, and then responding accordingly in the game's logic*
- **updating the game**
  - *updating, modifying anything in the game that needs a change*
    - e.g. graphics, music, interaction &c.
  - *a character moving - need to work out where they might be moving &c.*
  - *characters, elements in the game collide*
    - what happens when they collide? &c.
    - i.e. responding to changes in state and modifying a game...
- **rendering to the screen**
  - *drawing modifications, updates, &c. to the screen*
  - *we've worked out what needs to change*
  - *we're now drawing (rendering) those changes*
- if using FPS for game type
  - *may also need to consider how many times this game loop repeats*
  - *i.e. frames per second that this loop repeats*
  - *FPS may be important to ensure game is not running too fast or too slow*

Python and Pygame - game template

part 5 - add game loop

- we'll need to add a game loop to control and manage this pattern
  - *we're listening for inputs, events...*
  - *then updating the game*
  - *and finally rendering any changes for the user*
- we can add a standard `while` loop as a our primary game loop

```
# define boolean for active state of game
active = True
# create game loop
while active:
    # 'processing' inputs (events)
    # 'updating' the game
    # 'rendering' to the screen
```

- loop will follow defined pattern
  - *processing inputs (events)*
  - *updating the game*
  - *rendering to the screen*
- boolean `active` allows us to monitor the active state of the game loop
  - *as long as the value is set to `True` it will keep running*
  - *update this value to `False` and we may exit the game loop*
  - *we'll also see other ways of handling this exit...*

Python and Pygame - game template

part 6 - process inputs

- as the game is running
  - *a player should be able to interact with the game window*
  - *e.g. clicking the exit button, perhaps a keyboard, mouse or controller button...*
- if we consider the nature of a `while` loop
  - *we may initially see an issue with the underlying logic*
  - *e.g. the loop is either updating or rendering*
  - *what happens if a user clicks a button on the keyboard?*
- we need to be able to listen and record all events for our game
  - *regardless of the current executed point in the `while` loop*
- if not, only able to listen for events at the start of the loop
  - *as part of the processing logic*
- thankfully, Pygame has a solution for this issue

Python and Pygame - game template

part 7 - Pygame event tracking

- Pygame is able to keep track of each requested event
  - *from one executed iteration of the game loop to the next*
- it remembers each and every event
  - *as the the game's `while` loop executes the updating and rendering logic*
- as the `while` loop executes the *processing* logic
  - *we're able to check if there have been any new events*
- e.g. now add a simple `for` loop
  - *check for each and every event that Pygame has saved*

```
...
for event in pygame.event.get():
    ...
```

- start by checking for an event registered as clicking on the exit button
  - *a user request to close the current game window*

```
...
for event in pygame.event.get():
    # check for window close click
    if event.type == pygame.QUIT:
        # update boolean for running
        active = False
```

- checking for a saved event
  - *simply indicates the user wants to close the current game window*
- update the value of the boolean for the active game
  - *setting the value of the `active` variable to `False`*
  - *game loop, our `while` loop, will now exit*
- then add a call to quit Pygame at the end of our current Python file. e.g.

```
...
pygame.quit()
```

- game will now exit, and the Pygame window will close

Python and Pygame - game template

part 8 - double buffering

- as we start to render colours, lines, shapes &c. to our Pygame window
  - *need to be careful not to re-render everything for each update*
  - *if not, our game will become **very** resource intensive...*
- we can use an option known as **double buffering**
- in Pygame, this uses a concept of pre-drawing
  - *then rendering as and when the drawing is ready to be viewed by the player*
  - *drawing is flipped to show the rendering to the player*
  - *e.g. we can the following to our template*

```
...
# flip our display to show the completed drawing
pygame.display.flip()
```

- **n.b.** `flip` must be the last call after drawing
  - *if not, nothing will be displayed to the game's player*

Python and Pygame - game template

part 9 - monitor FPS

- *game loop* may also need to monitor and maintain setting for our game's FPS
- currently FPS set to 30 frames per second
- within the logic of our *game loop*

```
...
# check game loop is active
while active:
    # monitor fps and keep game running at set speed
    clock.tick(FPS)
...
```

- Pygame is now able to keep our game running at the defined frames per second
- as the loop runs
  - *it will always ensure that the loop executes the required 1/30 second*
- as long as the loop is able to *process*, *update*, and *render*
  - *within this defined time period of 30 fps, rendering will be smooth*
  - *if not, usually the update is taking too long*
  - *our game will run with lag, appear jittery to the player*
  - *may need to consider optimisation for code and logic...*

Python and Pygame - game template

part 10 - finish the template

- as we're only listening for the exit event on the game window
  - *we don't currently have any game content to update*
- our current template has set up a game window, and environment
  - *to test initial setup and initialisation*
  - *then allow a player to exit the game and window*
  - *e.g.*

```
...
# quit the Pygame window, exiting the game
pygame.quit()
...
```

Python and Pygame - game template

another example template

```
# import modules for pygame template
import pygame, sys

# variables for pygame
winWidth = 800
winHeight = 600

# variables for commonly used colours
BLUE = (0, 0, 255)

# initialise pygame settings and create game window
pygame.init()
window = pygame.display.set_mode((winWidth, winHeight))
pygame.display.set_caption("game template")

# define game quit and program exit
def gameExit():
    pygame.quit()
    sys.exit()

# create game loop
while True:
    # 'processing' inputs (events)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            gameExit()
    # 'updating' the game

    # 'rendering' to the window
    window.fill(BLUE)
    # 'flip' display - always after drawing...
    pygame.display.flip()
```

Python and Pygame - basic drawing

extra notes

Extra notes are available for drawing with Python and Pygame.

For example,

- drawing with `rect` to create various shapes
- working with RGB and colours
- drawing with `circle`, `ellipse`, & creating custom shapes

Notes include,

- drawing - basic
- colours

Python and Pygame

extras

- more Python and Pygame examples
  - *more drawing, animation, interaction and controllers...*
  - *game demo*
- latest extra notes on Python and Pygame include
  - *animation - colour scale*
  - *colours*
  - *control and move*
  - *drawing - basic*
  - *drawing - moving shapes*
  - *events - interaction*
  - *getting started*
- 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

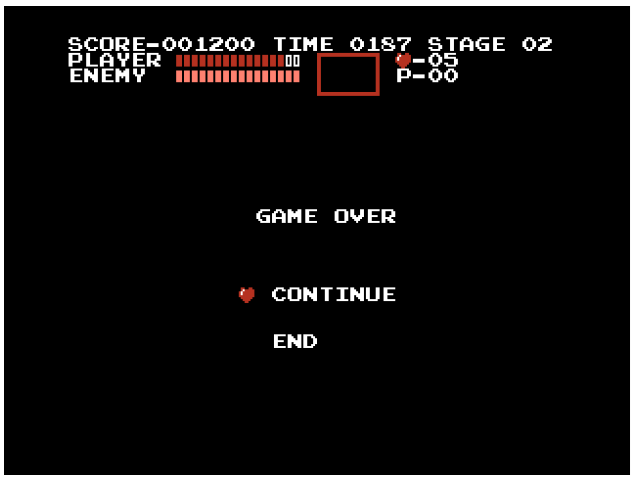
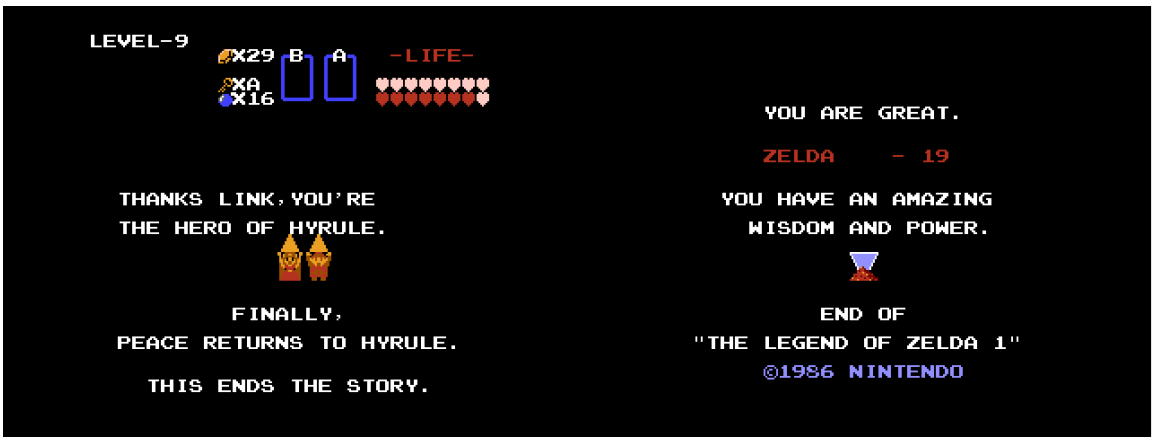
Super Mario Bros.	Castlevania
	

Image - Create a memorable ending

Legend of Zelda

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*

Python and Pygame - moving shapes

basic animation - to the right

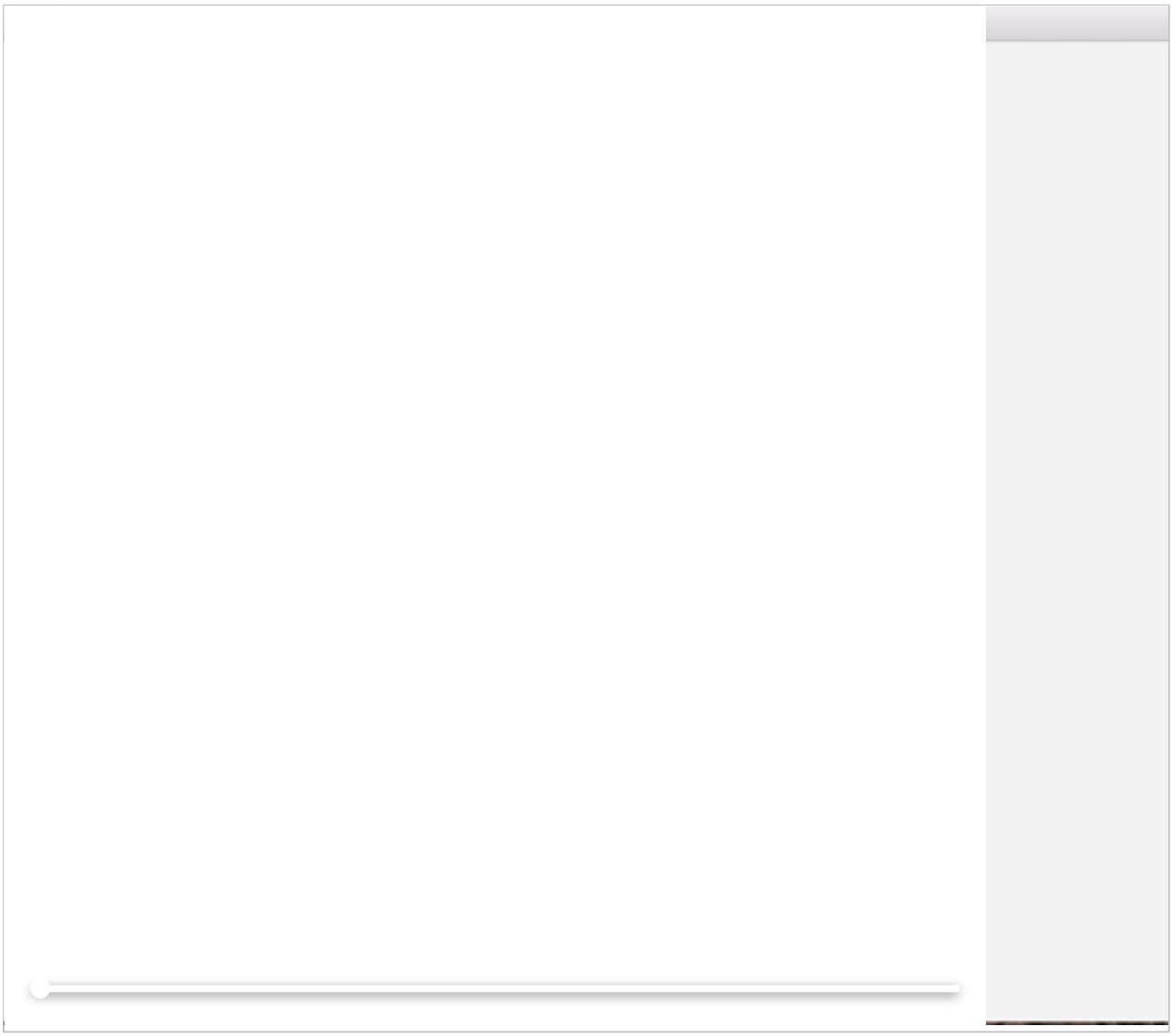
```
...
# rect coords...start at the centre
rectX = winWidth / 2
rectY = winHeight / 2

# create game loop
while True:
    # 'processing' inputs (events)
    ...
    # 'updating' the game
    # modify rectX by 4 pixels - higher creates impression of faster animation...
    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*

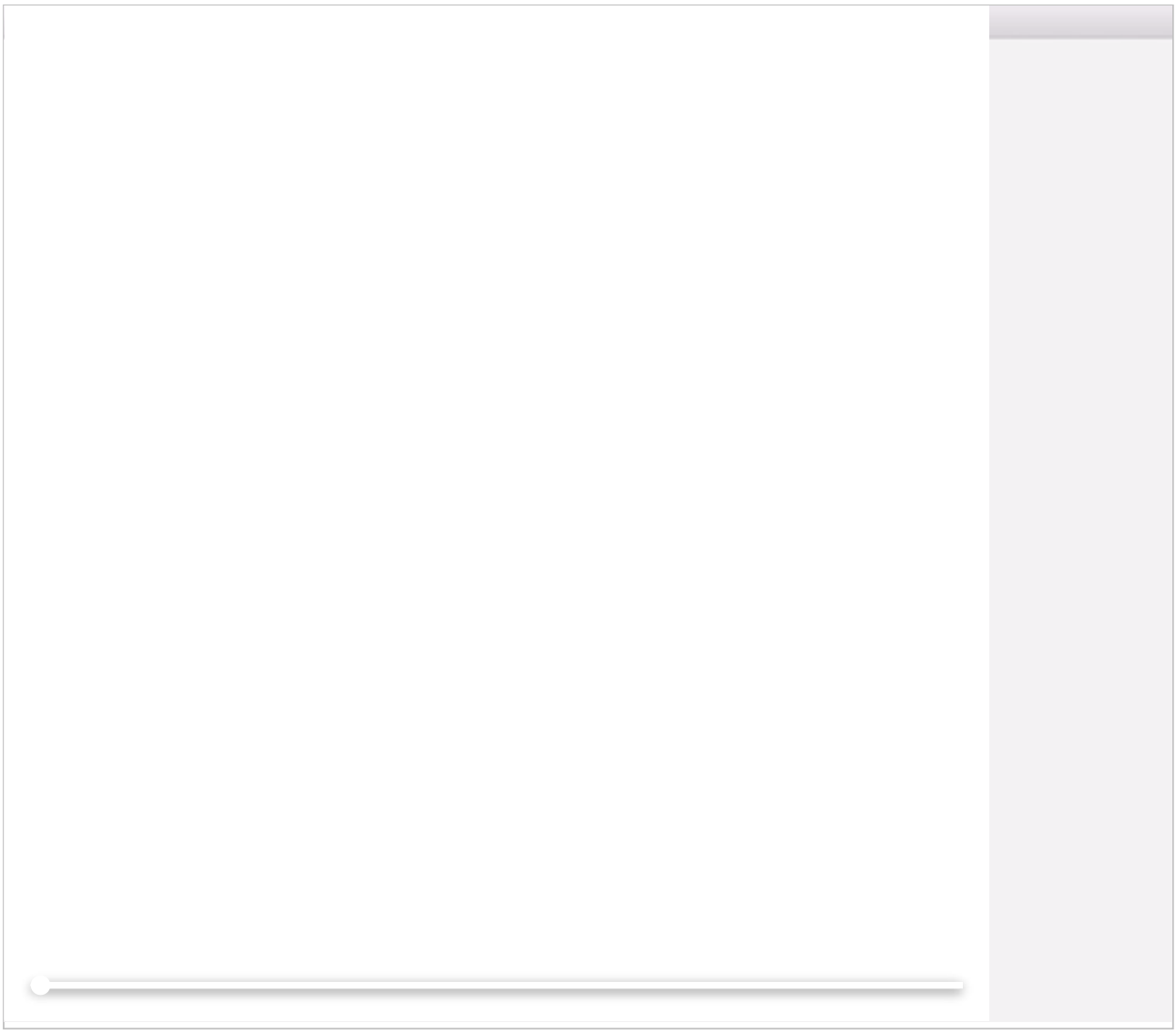
```
...
# modify rectX by 4 pixels
rectX -= 4
...
```

- 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
...
```

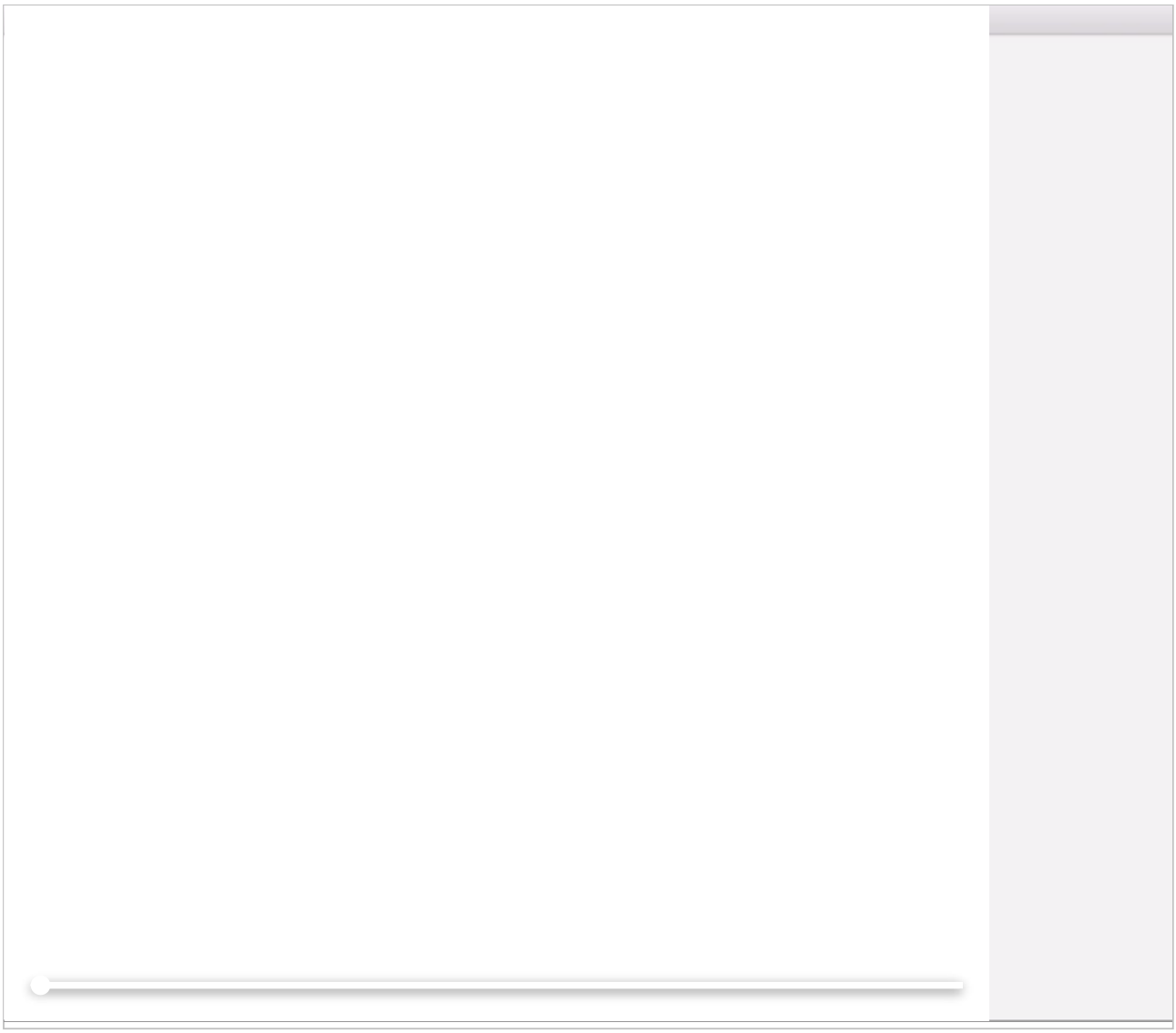
Video - Moving Shapes

basic animation - move to the left



Video - Moving Shapes

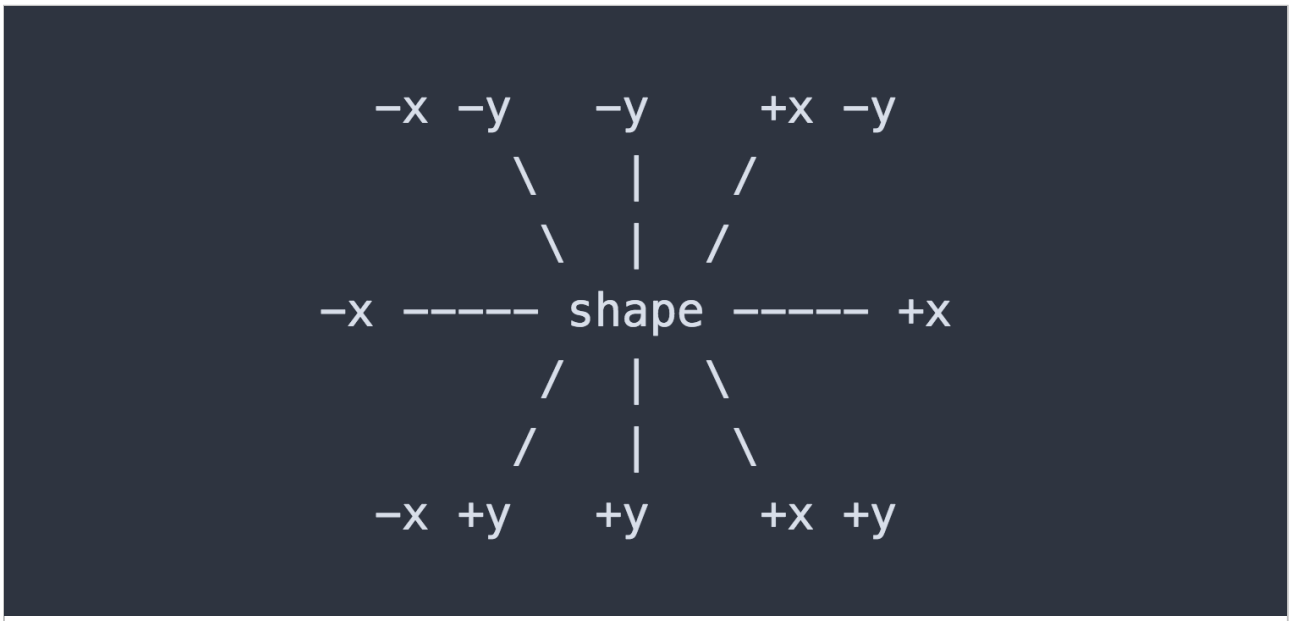
basic animation - move at an angle



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

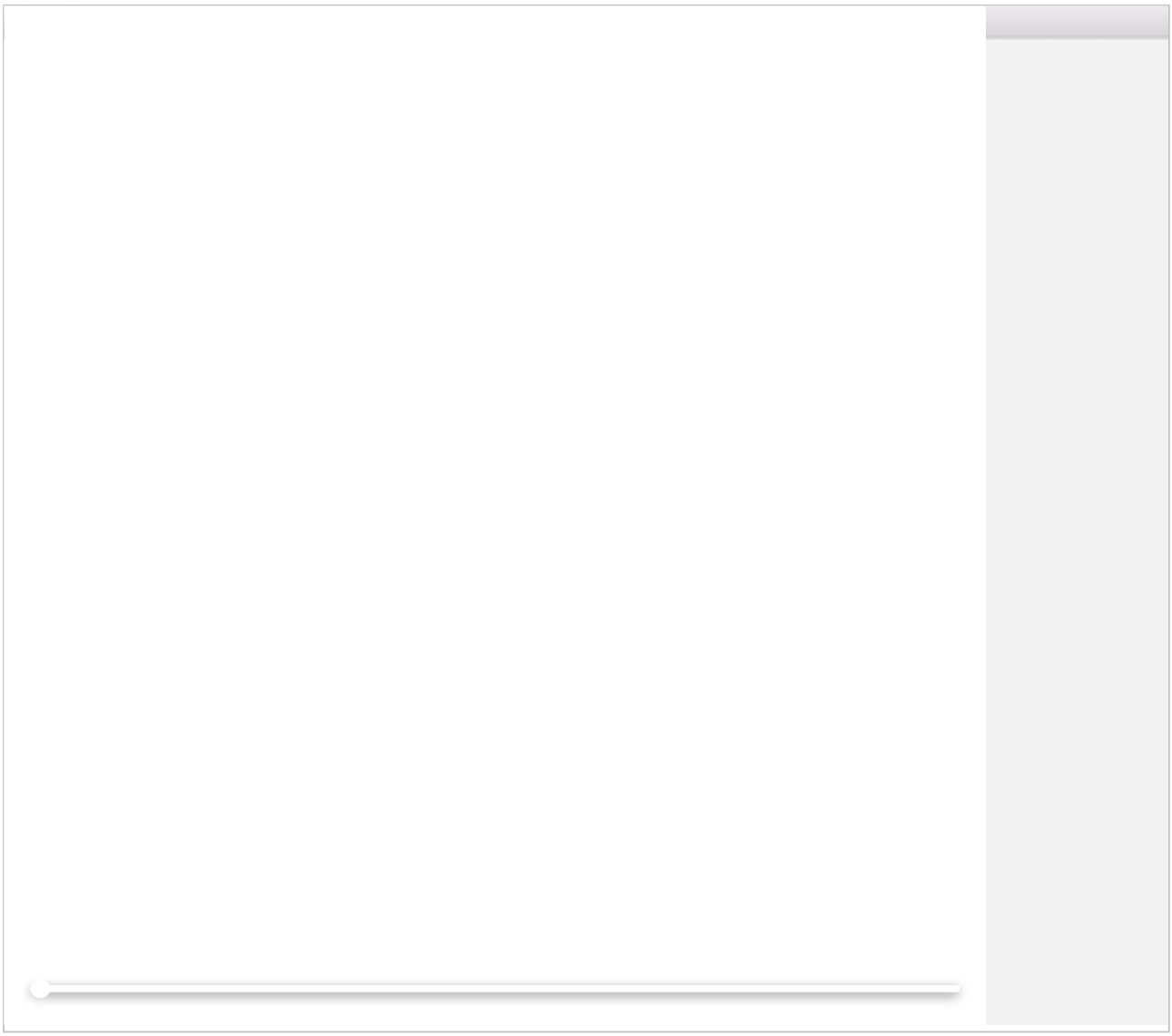
basic animation - check position

- 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 examples
  - *more drawing, animation, interaction and controllers...*
  - *game demo*
- latest extra notes on Python and Pygame include
  - *animation - colour scale*
  - *colours*
  - *control and move*
  - *drawing - basic*
  - *drawing - moving shapes*
  - *events - interaction*
  - *getting started*
- check course GitHub account for
  - *extra notes, various documents - notes*
  - *examples & templates - source*

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
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
- Proteus
- Zork - Downloads
- Zork - original version for PDP
- Zork 1 - Apple 2e version
- Zork 1 walkthrough - very useful

References

- Suits, B. *The Grasshopper: Games, Life and Utopia*. Broadview Press. 3rd Edition. 2014.
- Pygame
- pygame.event
- pygame.locals
- Wikipedia
- Draughts
- Space Invaders
- Zork

Videos

- Abzu trailer - YouTube