Comp 388/488 - Game Design and Development

Spring Semester 2019 - Week 6

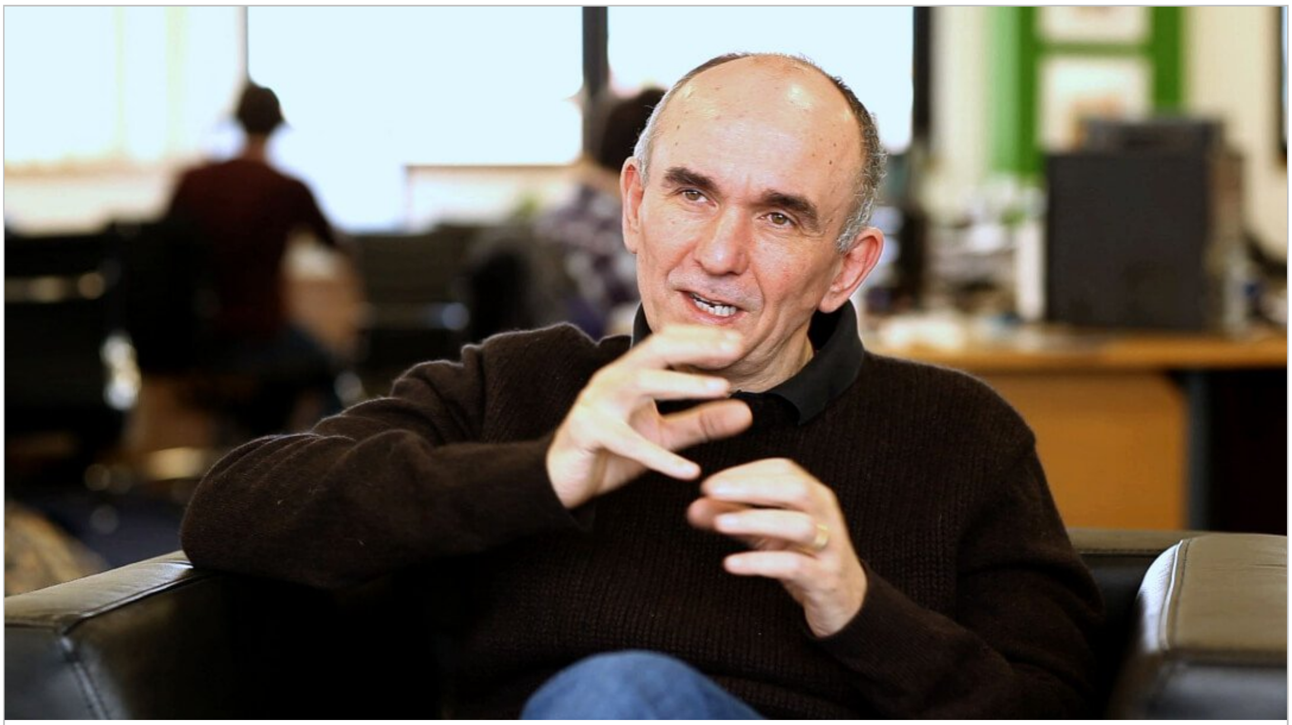
Dr Nick Hayward

Game designers

designer example - Peter Molyneux

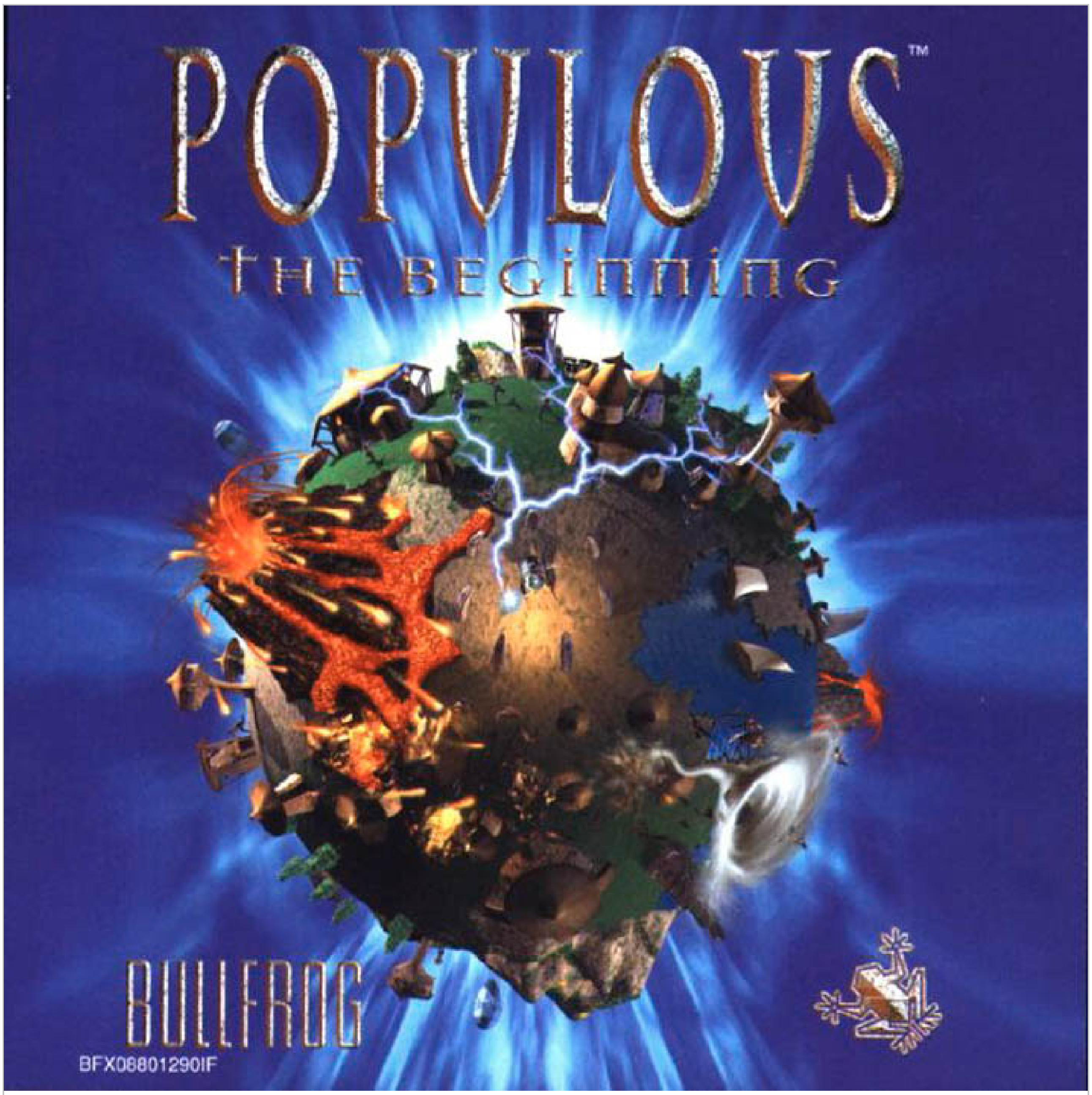
- well known example of a designer who pushed boundaries
  - *in particular, what we perceive as a game*
- breakthrough moment came with the design of the game **Populous**
  - *effectively created the* ***god*** *gaming genre*
- **Populous** was released in 1989 by his company **Bullfrog**
  - *sold over 4 million copies*
  - *best version originally released on the Commodore Amiga*
- **Black and White** game for Windows PCs released in 2001
  - *known for its unique design and gameplay*
  - *its overall depth and scope*
  - *renowned for its creatures' artificial intelligence*
  - *set a new Guinness World Record for its overall complexity*
- he also created game series such as
  - *Dungeon Keeper*
  - *Theme Park*
  - *Fable*
  - *The Trail*
  - ...

Image - Peter Molyneux



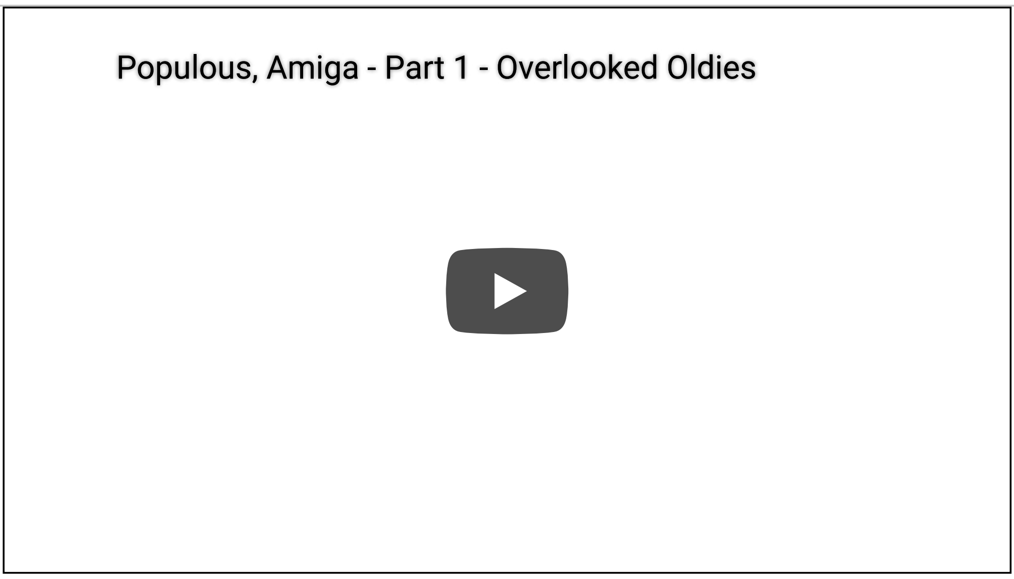
Peter Molyneux

Image - Populous - 1989



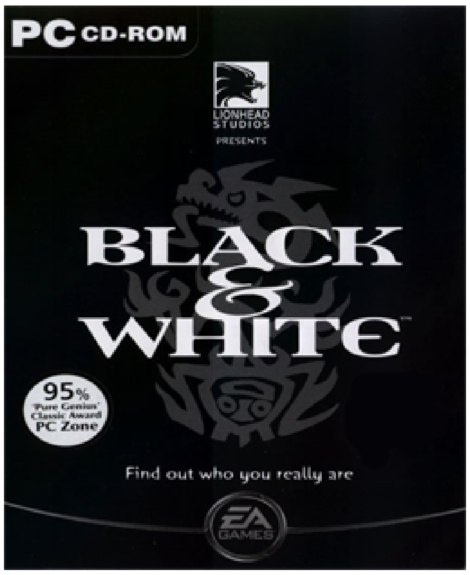
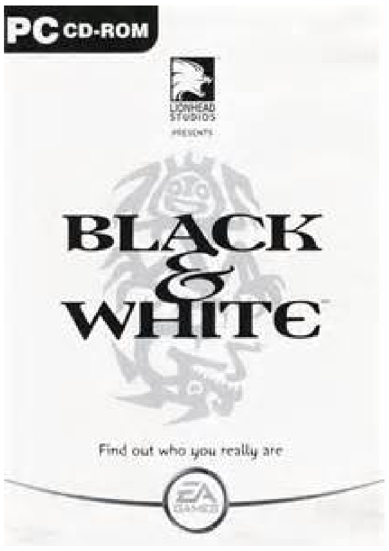
Populous cover

Video - Populous - Amiga

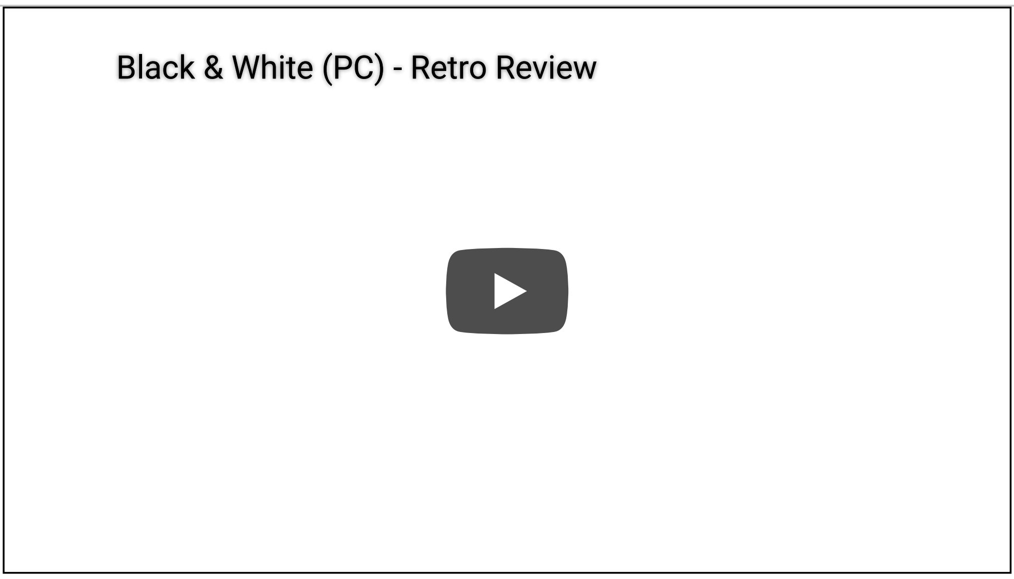

Populous, Amiga - Part 1 - Overlooked Oldies

Source - Populous on the Amiga, Youtube

Image - Black and White - 2001

Black cover	White cover
 PC CD-ROM LIONHEAD STUDIOS PRESENTS BLACK & WHITE 95% 'Pure Genius' Classic Award PC Zone Find out who you really are EA GAMES	 PC CD-ROM LIONHEAD STUDIOS PRESENTS BLACK & WHITE Find out who you really are EA GAMES

Video - Peter Molyneux's Black and White



Source - Black and White review, YouTube

Python and Pygame - moving shapes

basic animation - vertical - up

- move, and animate, our shapes using a vertical path
  - *from top to bottom, up and down*
- **move up**
  - *decrease or remove the `Y` value of a shape's position*
  - *e.g.simply remove 4 pixels per iteration of the game loop*

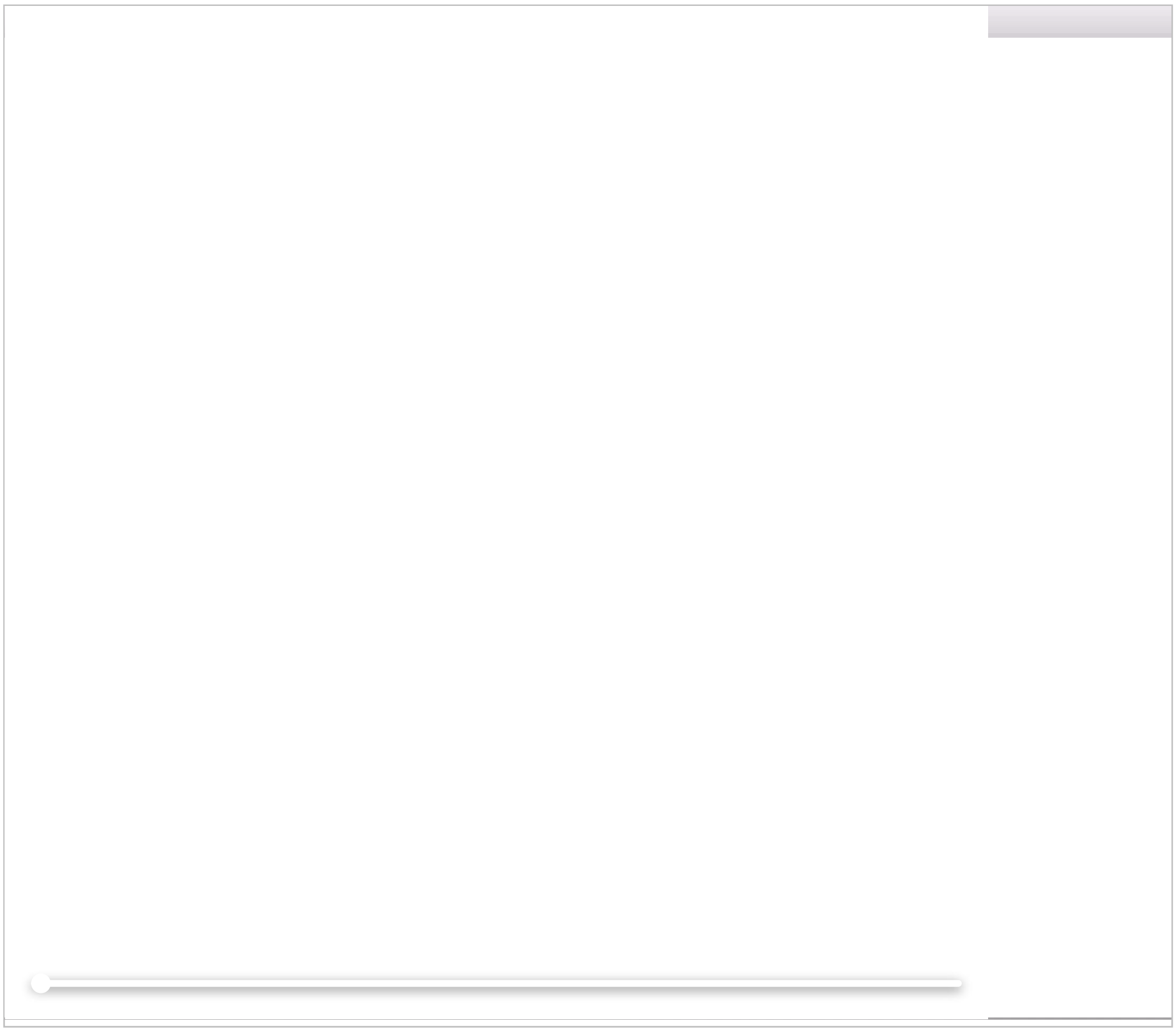
```
...
rectY -= 4
...
```

- detect our shape's position relative to the top edge of the window
  - *then animate it up from the bottom*

```
# check position of rectY and continue animation
if rectY < 0:
    rectY = winHeight
else:
    rectY -=4
```

Video - Moving Shapes

basic animation - move up



Python and Pygame - moving shapes

basic animation - vertical - down

- **move down**
  - *increase or add the Y value of a shape's position*
  - *e.g.simply add 4 pixels per iteration of the game loop*

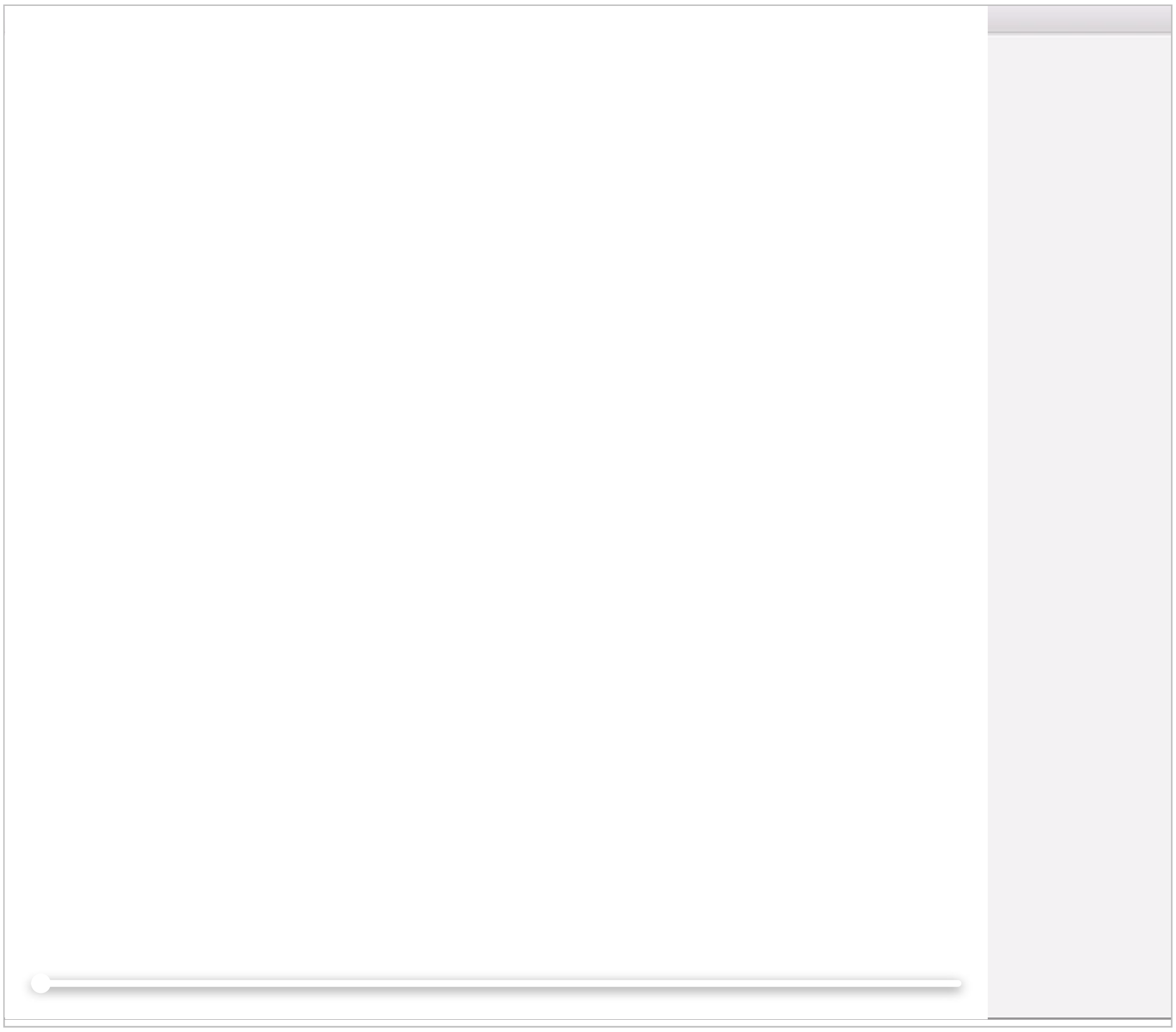
```
...
rectY += 4
...
```

- check as the shape leaves the game window
  - *continue animation from the top of the window*

```
# check position of rectY and continue animation
if rectY > winHeight:
    rectY = 0
else:
    rectY +=4
```

Video - Moving Shapes

basic animation - move down



Python and Pygame - events

intro

- detect interaction events with Pygame
  - *then allow a player to control shapes, animations, &c*
- as the *game loop* is executed
  - *Pygame keeps a record of interaction events for the game window*
- regardless of the execution point of the *game loop*
  - *e.g. update or drawing...*
  - *each event is added to `events`...where applicable*
- we may then check `events` to see if a particular key has been pressed
  - *or perhaps a controller button clicked*
- we start by importing `pygame.events`
  - *may be used with the keyboard, mouse, &c. events...*

```
import pygame.event
```

Python and Pygame - events

keyboard

- detect interaction events for keys pressed by a player whilst the Pygame window is running
- if we wanted to check for a given key press
  - *we may add a generic listener for `KEYDOWN`, `KEYUP`, `KEY_ESCAPE`....*

```
...
# check keyboard events - keydown
if event.type == pygame.KEYDOWN:
...
```

- then check a specific key event relative to keydown
  - *perhaps a player request to move a shape left or right...*

```
# check keyboard events - keydown
if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_LEFT:
        leftDown = True
    if event.key == pygame.K_RIGHT:
        rightDown = True
```

- we may also check specific lettered keys
  - *such as the `f` character, again as part of a key press down*

```
if event.key == pygame.K_f:
```

- simply listening for a key press on the `f` key on the player's keyboard
  - *perhaps allowing a player to toggle the game window fullscreen*
- many more examples listed on the Pygame website,
  - *Pygame - key*

Python and Pygame - events

keyboard - control shape left to right - part 1

- create a standard listener for an interaction event
  - *e.g. a keyboard event*
- we may then move our shape using one of 4-points on a coordinate plane
  - *left, right, up, and down*
- then check a specific key event relative to keydown
  - *perhaps a player request to move a shape left or right*
- on the `KEYDOWN` event, we update the boolean value for the requested key

```
# check keyboard events - keydown
if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_LEFT:
        leftDown = True
    if event.key == pygame.K_RIGHT:
        rightDown = True
```

- then reset it to `FALSE` on the `KEYUP` event,

```
# check keyboard events - keyup
if event.type == pygame.KEYUP:
    if event.key == pygame.K_LEFT:
        leftDown = False
    if event.key == pygame.K_RIGHT:
        rightDown = False
```

Python and Pygame - events

keyboard - control shape left to right - part 2

- we can use the set *boolean* value to modify the animation of a shape, e.g.

```
...
# event variables - keyboard
leftDown = False
rightDown = False
# some rect variables
rectSpeed = 4.0
...
# move left
if leftDown:
    # check shape doesn't exit window to left
    if rectX > 0.0:
        rectX -= rectSpeed
# move right
if rightDown:
    # check shape doesn't exit window to right
    if rectX + rectSize < winWidth:
        rectX += rectSpeed
...
```

- we're checking the boolean value for left or right key down
  - *if set to true, i.e. the player has pressed the key down*
  - *we can then check the shape's x coordinate position*
- check either the left or right side of the game window relative to the key pressed
- then, either increment or decrement the shape's x coordinate
  - *by the set speed for our animation*

Video - Interaction Events

keyboard - control shape - left to right

Python and Pygame - events

keyboard - control shape - up and down

- also use such interaction events to animate our shape up or down the screen
  - *set a boolean value to `TRUE` or `FALSE`*
  - *relative to the `KEYUP` or `KEYDOWN` event*
- then, we can animate our shape up and down the game window

```
...
# event variables - keyboard
upDown = False
downDown = False
# some rect variables
rectSpeed = 4.0
...
# move up
if upDown:
    # check shape doesn't exit window at top
    if rectY > 0.0:
        rectY -= rectSpeed
# move down
if downDown:
    # check shape doesn't exit window at bottom
    if rectY + rectSize < winHeight:
        rectY += rectSpeed
```

Video - Interaction Events

keyboard - control shape - up and down

Python and Pygame - events

keyboard - control shape - 8-point move

- in addition to the standard left, right, up, and down directions...
  - *combine these events to allow a user to move a shape in a diagonal direction*
- a player may simultaneously press `KEYDOWN` on both up and right
  - *allows a player to move a shape at a 45 degree angle*

```
...
# move up
if upDown:
    # check shape doesn't exit window at top
    if rectY > 0.0:
        rectY -= rectSpeed
# move right
if rightDown:
    # check shape doesn't exit window to right
    if rectX + rectSize < winWidth:
        rectX += rectSpeed
```

- a player may also use other available combinations to move the shape
  - *at one of 4 available angles of 45 degrees...*

```

        -x -y         +x -y
             \       /
              \     /
               shape
              /     \
             /       \
        -x +y         +x +y
```

Video - Interaction Events

keyboard - control shape - 8-point move

Python and Pygame - events

keyboard - control shape - jump - part 1

- to make a shape *jump*
  - *we may start by defining a useful boolean variable `shapeJump`*
- then simply update this value
  - *defines whether the character is jumping or not*
- also define a default pixel height for the jump itself
  - *simply defining how far to move the shape up the game window*

```
jumpHeight = 30.0
```

- then, we can add a listener for the defined key
  - *e.g. we might simply use the obvious UP directional arrow on our keyboard*

```
...
# check keyboard events - keydown
if event.type == pygame.KEYDOWN:
    # check for directional UP key
    if event.key == pygame.K_UP:
        if not shapeJump:
            shapeJump = True
            shapeJY += jumpHeight
...
```

- we're listening for the standard player `KEYDOWN` event
  - *then the actual directional UP key event*
- check the boolean value of the variable `shapeJump`
  - *update to `True` if the shape is not already jumping*
- then, incrementally update value of the shape's requested jump Y value, `shapeJY`

Python and Pygame - events

keyboard - control shape - jump - part 2

- to make the shape jump, or effectively move up the screen per iteration of the game loop
  - *we can define a function to handle this jump, `jump()`*

```
def jump():
    global shapeY, shapeJY, shapeJump

    # check if shape in air - use gravity to descend
    if shapeJump == True:
        shapeY -= shapeJY
        print("in the air %8.2f" % (shapeJY))
        shapeJump = False
```

- check the output of the jump up the screen
  - *e.g. printing the formatted float to the terminal.*
- if you run this example...
  - *you'll notice that the shape will keep jumping as the player presses the UP directional key*
  - *well beyond the bounds of the top of the game window*
- Pygame window needs to scroll...

Video - Interaction Events

keyboard - control shape - jump, jump, jump...

Python and Pygame - events

keyboard - control shape - jump and fall - part 1

- we could make the shape move down the window
  - *e.g. by listening for an explicit key press on the* ***DOWN*** *directional key*
- it's more natural, and expected behaviour, to allow our shape to fall
  - *after the player has pressed the* ***UP*** *directional key*
  - *allowing our shape to jump, and then fall*
  - *fall with a real-world behaviour of* ***gravity***
- to make it fall, we need to check that the shape is *in the air*
- then gradually modify gravity to lower the shape
  - *lower to the original starting position in the Pygame window*

Python and Pygame - events

keyboard - control shape - jump and fall - part 2

code example

```
def jump():
    global shapeY, shapeJY, shapeJump, gravity
    # check upward speed > 1.0
    if shapeJY > 1.0:
        # gradually decrease upward speed to less than 1.0
        shapeJY = shapeJY * 0.9
    else:
        # less than 1.0, reset to 0.0 to allow shape to fall
        shapeJY = 0.0
        # stop jump
        shapeJump = False

    # check if shape in air - use gravity to descend
    if shapeY < winHeight - shapeSize:
        shapeY += gravity
        gravity = gravity * 1.1
    else:
        shapeY = winHeight - shapeSize
        gravity = 1.0

    shapeY -= shapeJY
```

Python and Pygame - events

keyboard - control shape - jump and fall - part 3

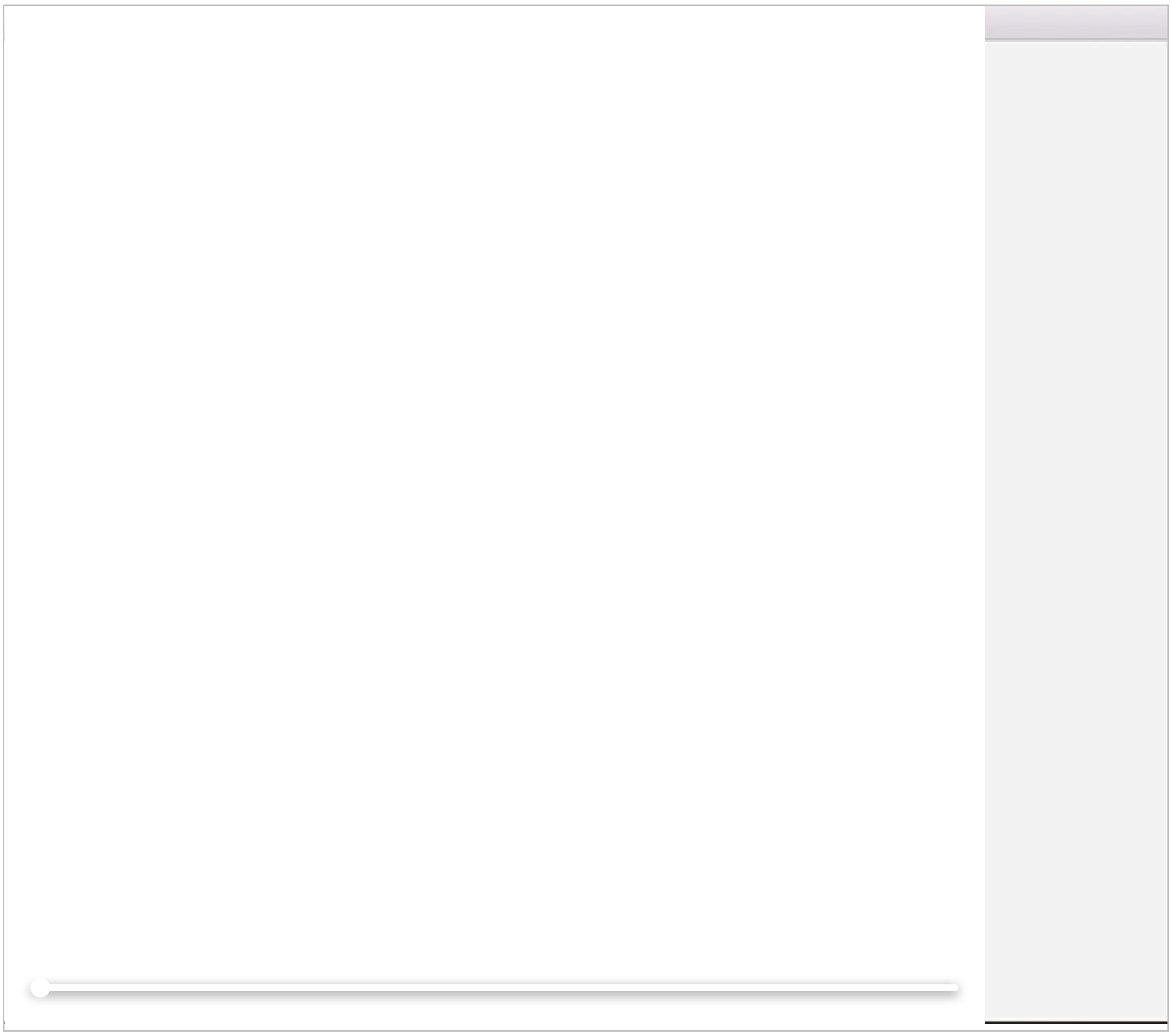
code example outline

- in the previous code example
  - *start by checking whether the shape is still moving up the screen*
  - *effectively if the jump is still in progress*
- whilst the upward speed of the shape is still above `1.0`
  - *gradually start to decrease the speed*
  - *it will eventually reach a limit for the jump*
- faster we decrease this upward motion
  - *the shorter the shape will appear to jump*
- also negates the overall effect of the value of the variable `jumpHeight`
  - *now has less iterations of the game loop to move the shape up the screen*
- need to check if the shape is actually moving up the screen
  - *or effectively in the **air** for the jump*
- if not, then the shape will simply come to a halt as it rises up the screen
  - *due to the decrease in upward speed and motion*
- we need to add the perception of **gravity** to the shape's motion
  - *whilst the shape appears to be in the **air**, or jumping up the screen*
  - *start to add the number of pixels we define for the variable **gravity***
  - *add pixels to our shape's upward movement*
- as the shape starts to fall down the game window
  - *slowly increase the value of the `gravity` variable*
  - *helps to suggest a realistic downward fall*
- if not, the jump and fall will not be timed correctly
  - *a player will perceive the shape's fall as very slow*

- *the fall will seem unrealistic, as though the gravity is too low...*

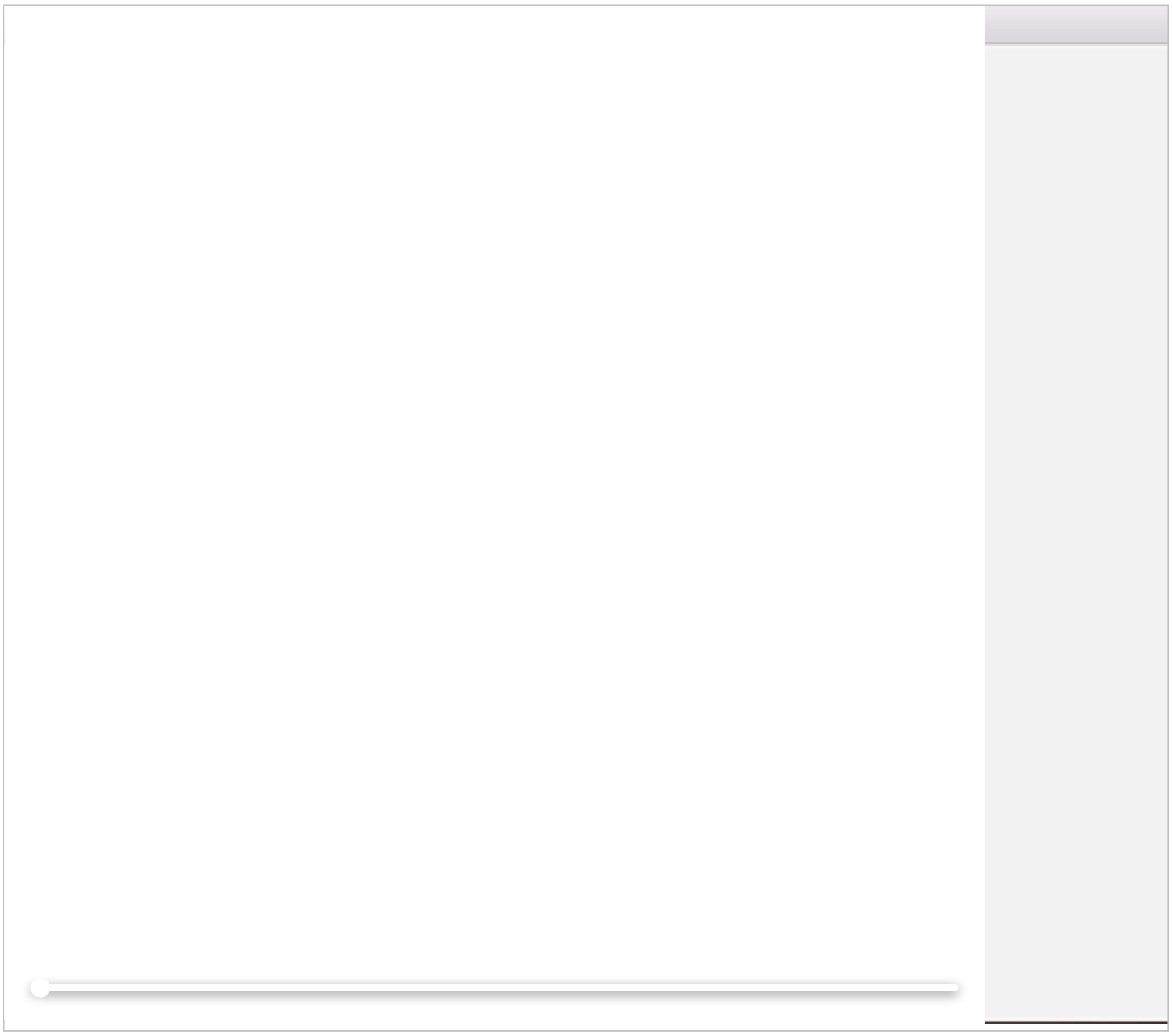
Video - Interaction Events

keyboard - control shape - jump and freeze



Video - Interaction Events

keyboard - control shape - jump and fall slowly



Video - Interaction Events

keyboard - control shape - jump and fall with gravity

Python and Pygame - events

keyboard - control shape - move, jump... - part 1

- now combine moving a shape horizontally, vertically, and jumping
  - *create a shape that a player can move and control freely in the Pygame window*

```
def move():
global shapeX, shapeY, shapeRX, shapeJY, shapeJump, gravity

# move left
if leftDown:
    # check shape not exit window to left
    if shapeX > 0.0:
        shapeX -= shapeSpeed
# move right
if rightDown:
    # check shape not exit window to right
    if shapeX + shapeSize < winWidth:
        shapeX += shapeSpeed

# check upward speed > 1.0
if shapeJY > 1.0:
    # gradually decrease upward speed to less than 1.0
    shapeJY = shapeJY * 0.9
else:
    # less than 1.0, reset to 0.0 to allow shape to fall
    shapeJY = 0.0
    # stop jump
    shapeJump = False

# check if shape in air - use gravity to descend
if shapeY < winHeight - shapeSize:
    shapeY += gravity
    gravity = gravity * 1.1
else:
    shapeY = winHeight - shapeSize
    gravity = 1.0

shapeY -= shapeJY
```

- move function combines horizontal movement with a vertical jump
  - *player can now make the shape move from left to right*
  - *and jump at the same time*

Python and Pygame - events

keyboard - control shape - move, jump... - part 2

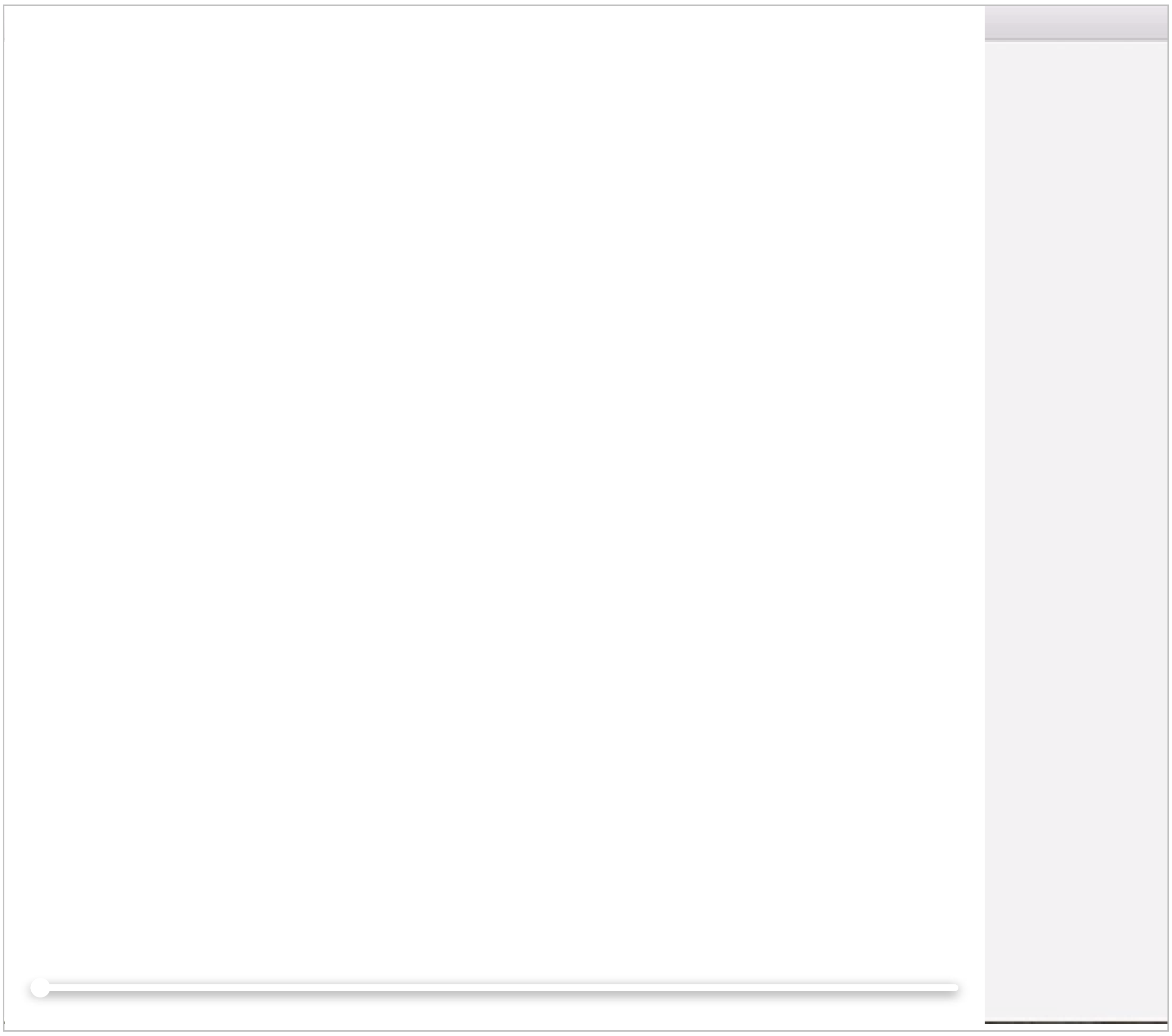
- update the **game loop** to include required listeners and handlers for horizontal movement
  - *add the required listener for KEYUP*
    - stop our shape from continuously moving right or left
- shape can now walk and jump across the game window

```python
# create game loop
while True:
    # set clock
    #msElapsed = clock.tick(max_fps)
    #print(msElapsed)
    # 'processing' inputs (events)
    for event in EVENTS.get():
        # check keyboard events - keydown
        if event.type == pygame.KEYDOWN:
            # check for directional - LEFT and RIGHT
            if event.key == pygame.K_LEFT:
                leftDown = True
            if event.key == pygame.K_RIGHT:
                rightDown = True
            # check for directional - UP
            if event.key == pygame.K_UP:
                if not shapeJump:
                    shapeJump = True
                    shapeJY += jumpHeight
            # check for ESCAPE key
            if event.key == pygame.K_ESCAPE:
                gameExit()

        # check keyboard events - keyup
        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT:
                leftDown = False
            if event.key == pygame.K_RIGHT:
                rightDown = False
```

Video - Interaction Events

keyboard - control shape - move and jump



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

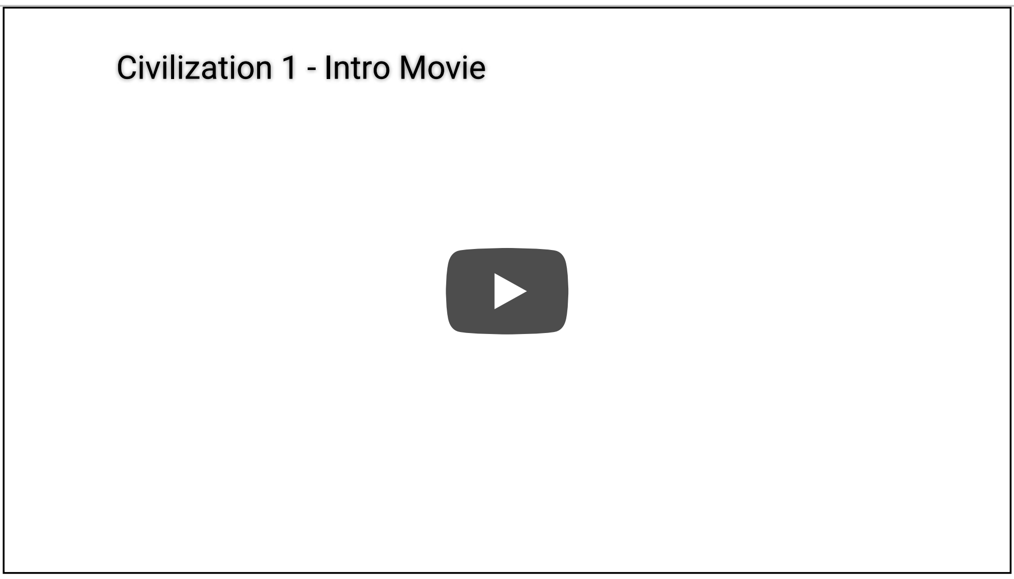
Games and Ideas

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

Animal Crossing (Nintendo Gamecube) - Retro Video Game...

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 - Sprites

intro

Please consult the extra notes on Pygame Sprites,

- sprites - intro

resources

- notes = sprites-intro.pdf
- code = basicsprites1.py

Python and Pygame - Sprites

image import

Please consult the extra notes on Pygame Sprites,

- sprites - set image

resources

- notes = sprites-set-image.pdf
- code = basicsprites2.py

Image - Image Sprite

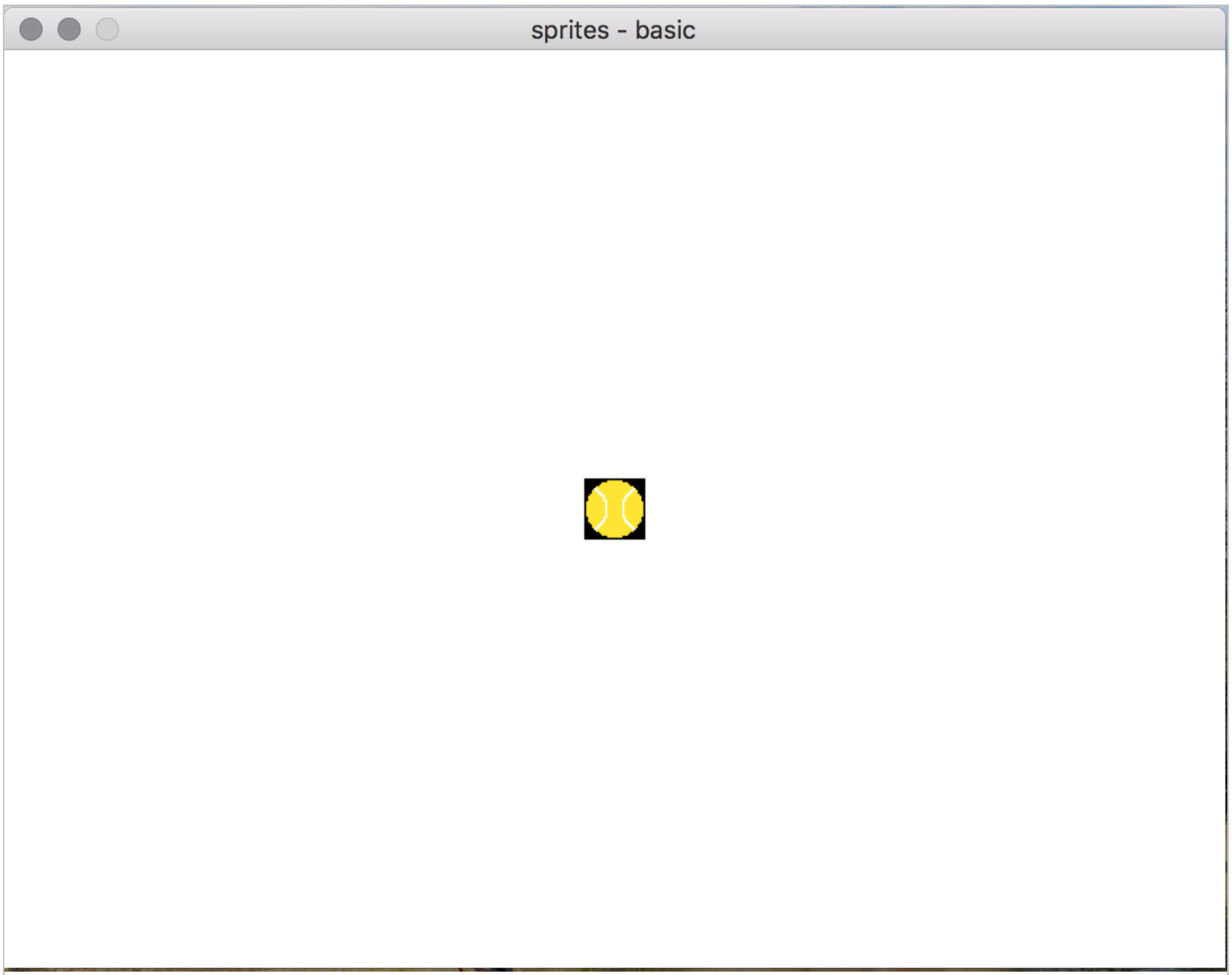


Image sprite

Image - Image Sprite

add transparency

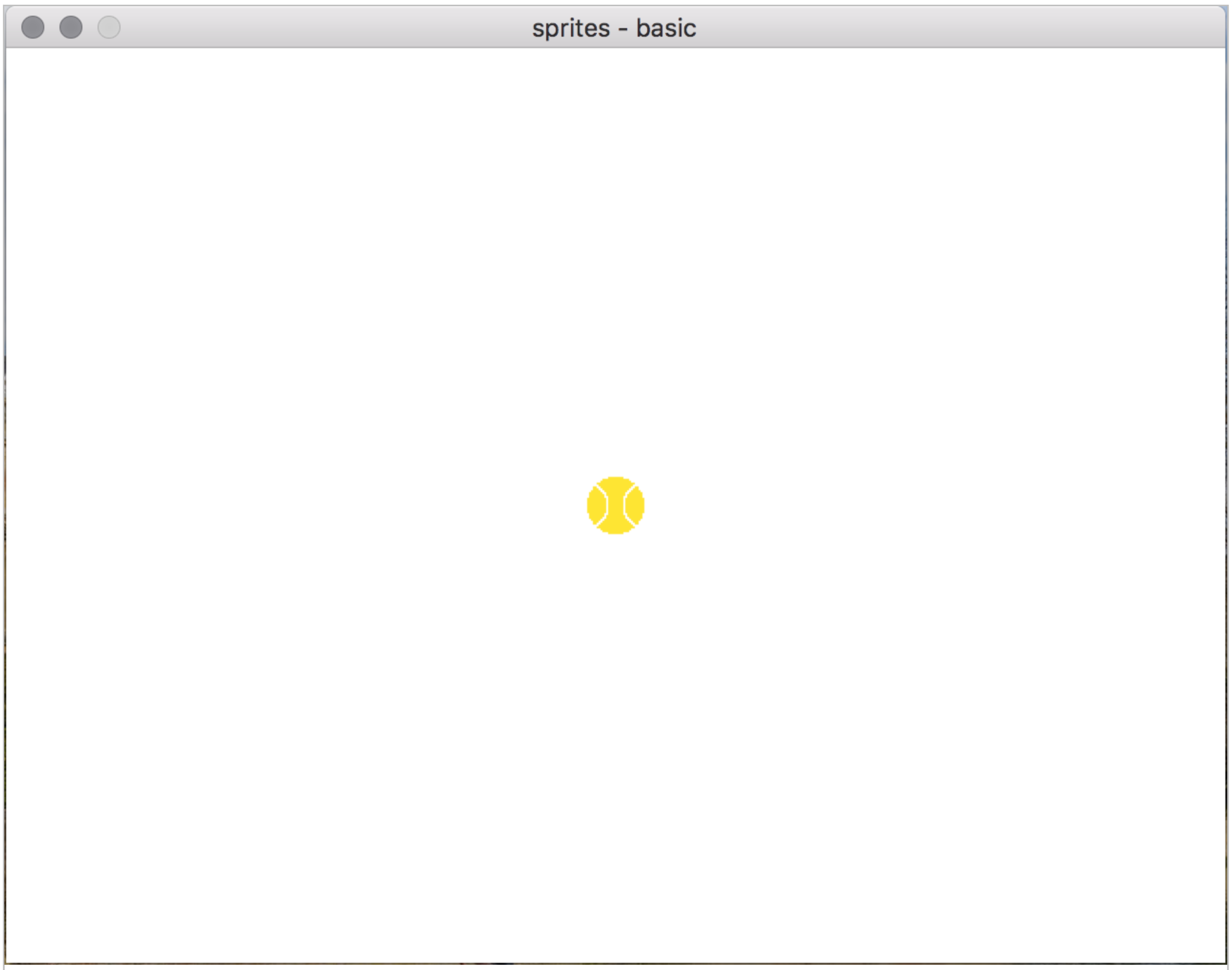
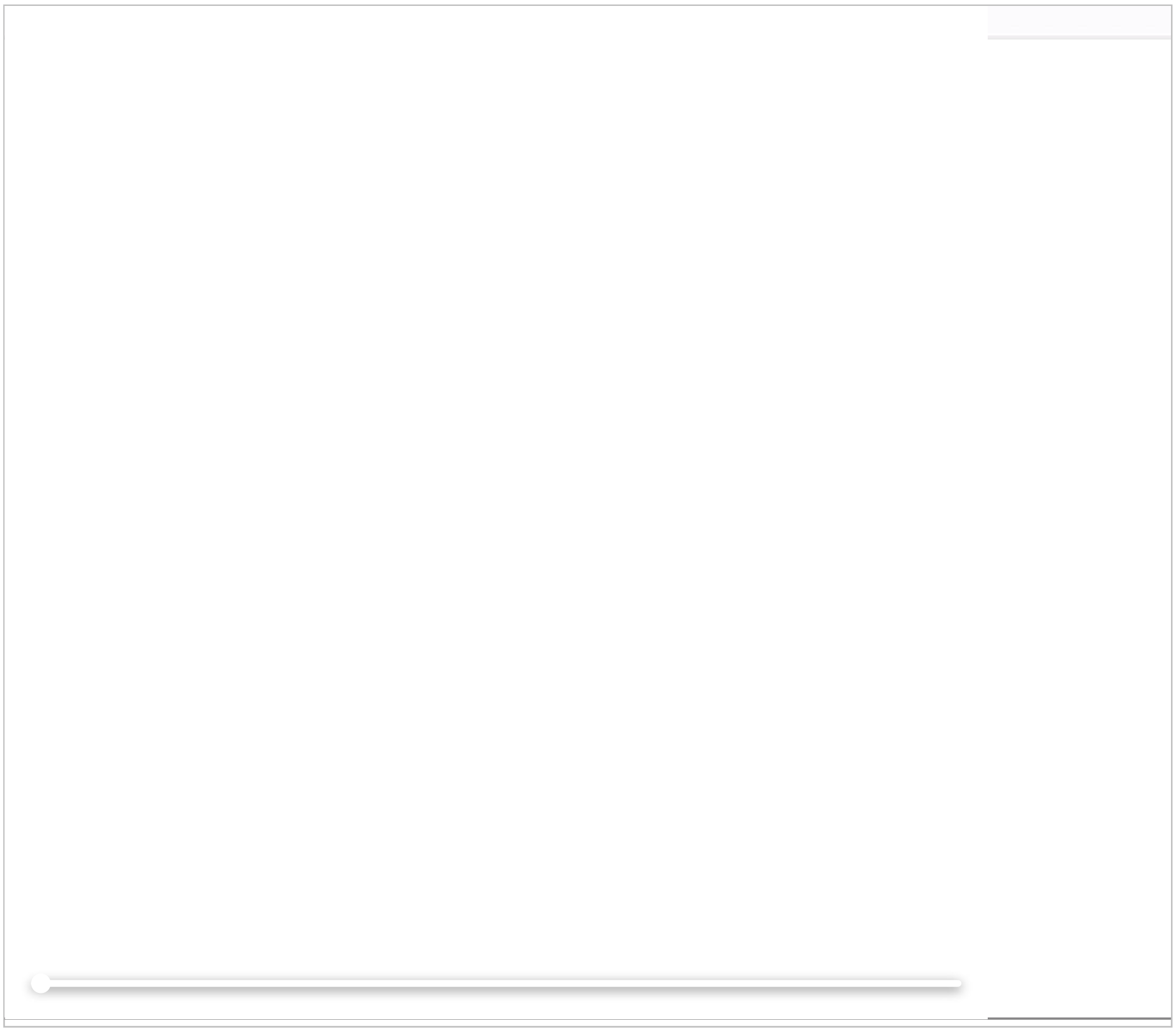


Image sprite - transparent background

Video - Image Sprite

bouncing ball



Python and Pygame - Sprites

control and move, add events...

Please consult the extra notes on Pygame Sprites,

- sprites - control

resources

- notes = sprites-control.pdf
- code = basicsprites3.py

game example

- shooter0.1.py - move & control

Video - Shooter 0.1

move & control

Demos

- pygame sprites - basic
  - *basicsprites1.py*
  - *basicsprites2.py*
  - *basicsprites3.py*
  - *basicsprites4.py*
  - *basicsprites5.py*
  - *basicsprites6.py*
- pygame graphics and sprites
  - *graphicssprites1.py*

Games

- Animal Crossing
- Black and White
- Civilization series
- Populous
- Proteus

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.
- 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*
- Pygame
  - *pygame.event*
  - *pygame.key*
  - *pygame.locals*
- various
  - *God Game*
  - *Peter Molyneux*
  - *Populous*
  - *GameDev.net*
  - *Video Game Design Schools*

Videos

- Animal Crossing
- Black and White review - YouTube
- Populous on the Amiga - Youtube
- Sid Meier's Civilization, Youtube
- The Last Startfighter, YouTube