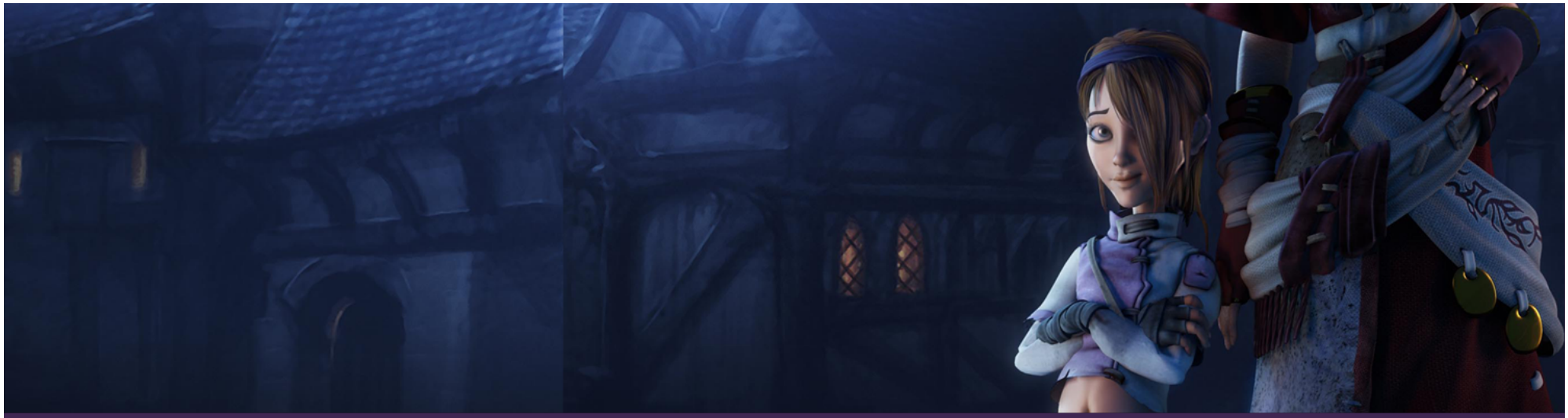


FALMOUTH
UNIVERSITY

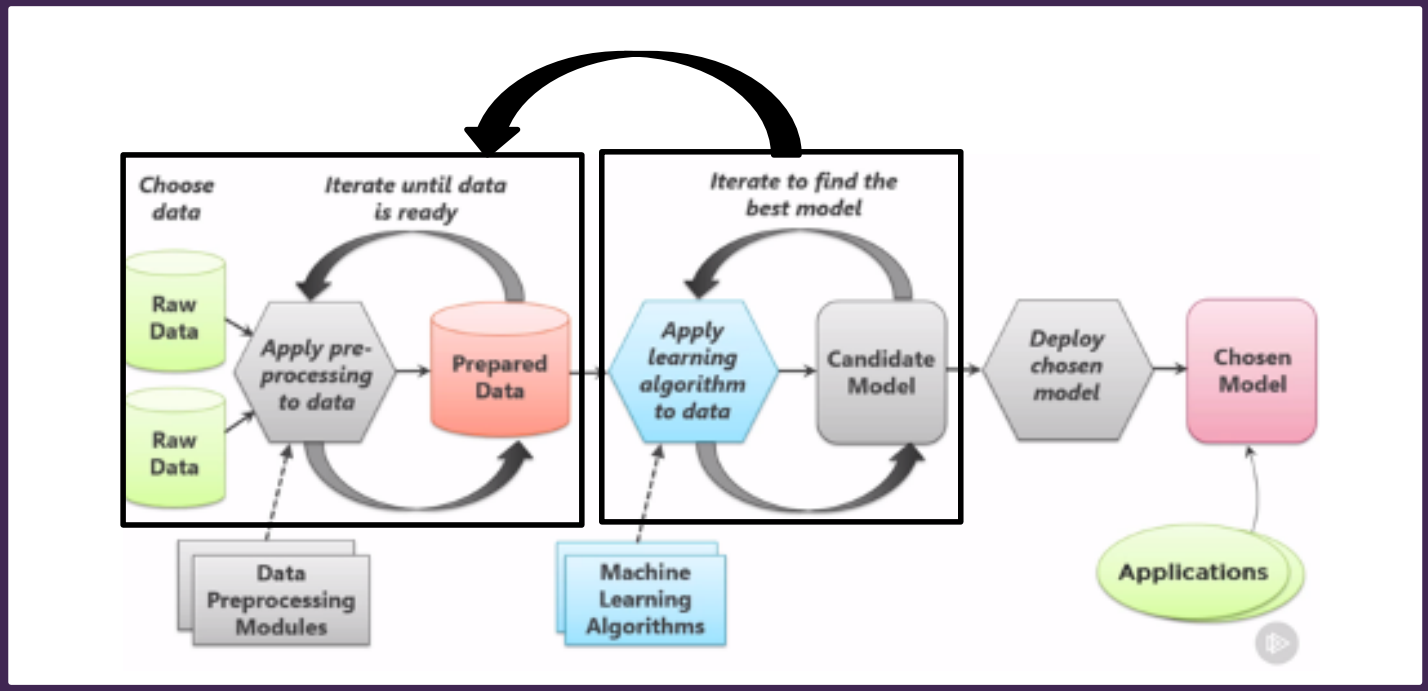
Lecture 5: Data Science – Machine Learning for gameplay

COMP704: Machine Learning
MSc Artificial Intelligence for Games

- Today's session:
  - Refactoring & Instrumenting Games for ML

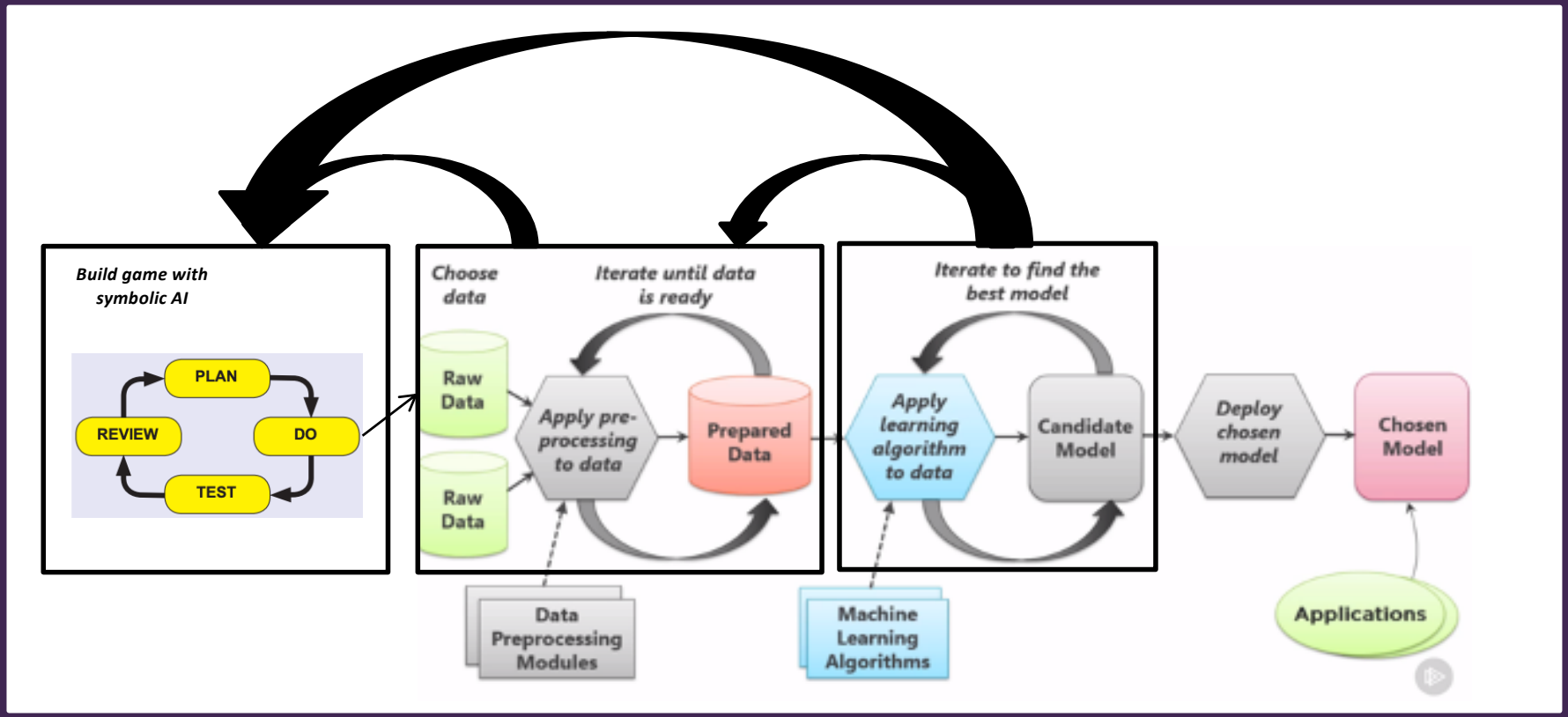
- Refactoring & Instrumenting Games for ML
  - What do you need to do to make applications that can ‘play’ using ML for AI?
    - Aka the assignment

- Refactoring & Instrumenting Games for ML
  - What do you need to do to make applications that can ‘play’ using ML for AI?
    - Aka the assignment



    - We’ve looked at this iterative model before

- Refactoring & Instrumenting Games for ML
  - Now we can add more complexity



- Refactoring & Instrumenting Games for ML
  - Now we can add more complexity
    - Added stage of ‘build a game’:
      - Building a game
      - Building a symbolic AI system
      - Capturing gameplay data
      - Feeding ML results into non-symbolic AI model
      - Rinse & repeat when it doesn’t work quite right
        - » Such is the nature of our black box AI

- Refactoring & Instrumenting Games for ML
  - Now we can add more complexity
    - Added stage of ‘build a game’:
      - Building a game
      - Building a symbolic AI system
      - Capturing gameplay data
      - Feeding into non-symbolic AI model
      - Rinse & repeat when it doesn’t work quite right
        - » Such is the nature of our black box AI
    - To a degree, these stages are interlinked
      - Your architectural & game play designs will be driven by what comes before and what follows each stage

- Refactoring & Instrumenting Games for ML
  - Over-arching consideration(s)
    - Need to be able to generate training (& test) data that will train an algorithm
      - This will depend on game type i.e. what kind of data can you extract from a game
        - » Does it need to be regression or classification
        - » Continuous or discrete
      - How much data do you need to collect
        - » Are there any significant use cases to consider?
      - How are you going to manage and store it

- Refactoring & Instrumenting Games for ML
  - Over-arching consideration(s)
    - Need to be able to generate training (& test) data that will train an algorithm
      - How will the data be generated
        - » What will your synthetic symbolic AI algorithms look like
        - » Will you augment this with real players?

- Refactoring & Instrumenting Games for ML
  - Over-arching consideration(s)
    - How will you feed the prediction data back into your game

- Refactoring & Instrumenting Games for ML
  - How to make a game that can capture data for ML
  - Breakout game
    - Not a particularly great pygame demo, but it could be a typical starting point
    - From a data PoV:
      - State of world (bricks & ball)
        - » Like the Mariflow world array
      - Either:
        - » Player position as regression
        - » Plyer movement as classifications (move left / right)

- Refactoring & Instrumenting Games for ML
  - Refactoring the game to make it work (better)
    - Fix dodgy collisions / gameplay issues
      - Ball can run along bat and off the screen
        - » Generally, want to only collision test ball when it is traveling down the screen
          - Current code doesn't allow this due to angle model
        - » May also be issues with ball colliding with two things at once
      - Hard to direct the ball off the bat
        - » In 'traditional' pong ball will come off at different angles depending on where on the bat it hits
          - Need to implement & use this to develop pong strategies (clear a column and get ball on top of bricks)
      - Game over on every miss
        - » Need to make the game run to level completion to collect more data

- Refactoring & Instrumenting Games for ML
  - Refactoring the game to make it work (better)
    - Make game play ‘headerless’
      - Currently, game will do one update per frame (16.6mS)
        - » From pygame.flip()
      - Removing rendering means we can run the game at 500-1000 fps or higher
        - » 16 x faster -> 16 x data collection
        - » Can do this without making the game ‘play faster’
        - » Can also run multiple instances of game and collect data remotely (HTTP server)

- Refactoring & Instrumenting Games for ML
  - Building symbolic AI
    - Where should the player go to hit the ball?
      - Easiest (and worst) solution
        - » Player.x = ball.x
        - » This will work, but wont produce great results
      - More interesting solutions require
        - » Where will the ball be when the player can hit it (forward projection)?
        - » Where can the player hit it to (prediction)?
        - » Where's the best place to hit the ball to (estimation)?

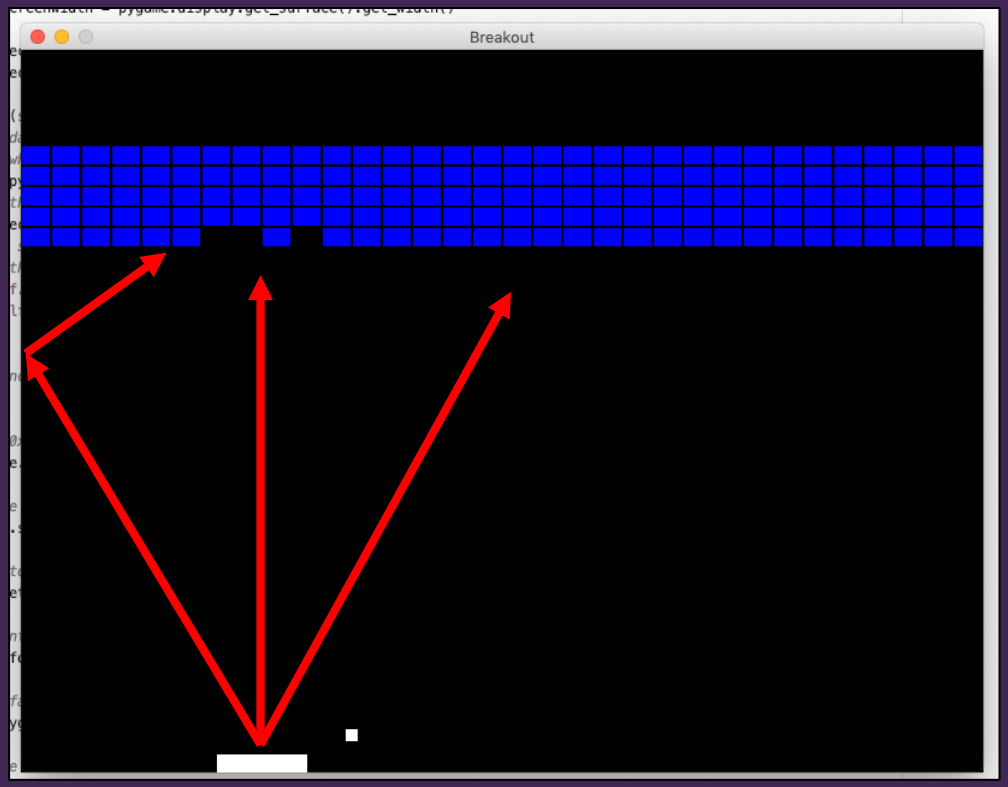
- Refactoring & Instrumenting Games for ML
  - Building symbolic AI
    - Where will the ball be when the player can hit it (forward projection)?
      - We know the algorithm for the ball's movement

```
# Change the position (x and y) according to the speed and direction
self.x += self.speed * math.sin(direction_radians)
self.y -= self.speed * math.cos(direction_radians)
```

      - Can have a copy of the ball's data (position & speed) and project where the ball will go
      - Best to do this under the bricks & when the ball is travelling towards the player (hence the refactoring)
        - » Ball will have a pretty consistent travel time in comparison to the upward journey
        - » Don't forget to include the bounces

- Refactoring & Instrumenting Games for ML
  - Building symbolic AI
    - Where can the player hit it to (prediction)?
      - This relies on refactoring the ball/player collider to make it a bit more predictable
      - Can use a similar approach to project balls up the screen to see what they hit

- Refactoring & Instrumenting Games for ML
  - Building symbolic AI
    - Where can the player hit it to (prediction)?



- Refactoring & Instrumenting Games for ML
  - Building symbolic AI
    - Where's the best place to hit the ball to (estimation)?
      - Give all the available bricks some value
      - Choose to hit whatever brick has the highest value
      - This is where we can look at breakout 'strategy'
        - » One strategy is to concentrate on a single column in order to break through to the top.
        - » Therefore, the deeper into a column bricks are, the higher value they will have
      - Also gives you an approach to actually clear levels

- Refactoring & Instrumenting Games for ML
  - Building symbolic AI
    - These three generic approaches form the basis of a lot of goal-oriented planning approaches
      - forward projection?
      - prediction?
      - estimation?
    - If you're not making breakout (and, of course, you aren't) look at these approaches for your symbolic AI

- Refactoring & Instrumenting Games for ML
  - Capturing Data
    - What data to capture
    - How to capture it
    - How to store it
    - How to process it for ML

- Refactoring & Instrumenting Games for ML
  - Capturing Data
    - What data to capture
      - You will have an idea of what data to capture
        - » Generally, look to capture game elements
        - » Keep in a raw format so you can process them off-line
          - Don’t want to lose capture data, unless your capture / symbolic AI changes
      - Will be refined with every piece of ML research you do
        - » See from house price data:
          - Granularity of data
          - Clustering / quantisation of data

- Refactoring & Instrumenting Games for ML
  - Capturing Data
    - How to capture it
      - What constitutes useful data?
        - » Mariflow captures at 15fps
          - This is different data to shooting games where fire events may occur at any time
        - » Do you want to capture failure data
          - What will break out data where paddle misses the ball give you?
      - May need to buffer data locally before you decide to commit it to your training data

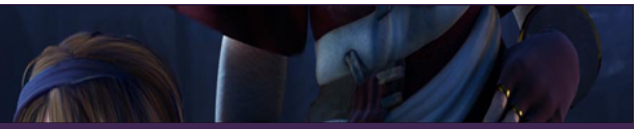
- Refactoring & Instrumenting Games for ML
  - Capturing Data
    - How to store it
      - If you're collecting 'lots' of data (multiple AI sessions)
        - » Look to manage through a database
          - Think about how sets of data will need to be managed and tagged to identify them
      - HTTP server example is a useful front end for this
        - » More so than lots of files

- Refactoring & Instrumenting Games for ML
  - Capturing Data
    - How to process it for ML
      - Saw this in last week's lecture
        - » Often data needs to be processed prior to ML process
        - » Keep source data in a source format
        - » Don't process during capture
          - If your processing is wrong (it's likely to be), you will lose all that data

- Refactoring & Instrumenting Games for ML
  - Feeding ML results into non-symbolic AI model
    - Need to think about how your game will work with prediction data
      - If you're classifying, can give the AI a player-like interface of 'key presses'
        - » As that will be the ML-algorithm output
      - Default breakout interface is mouse position
        - » No limitation on how far the player can move in one frame
          - Worth building those limitations back into the refactor to stop the AI from 'cheating'

- Refactoring & Instrumenting Games for ML
  - Rinse & repeat when it doesn't work quite right
    - All good blogging fodder (as is every step in this model)

- Tell me about your games …



- Do you have any questions for me?

- Workshop
  - This week will be assignment support :)
    - Bring in your work so far and we can work out how to take it further