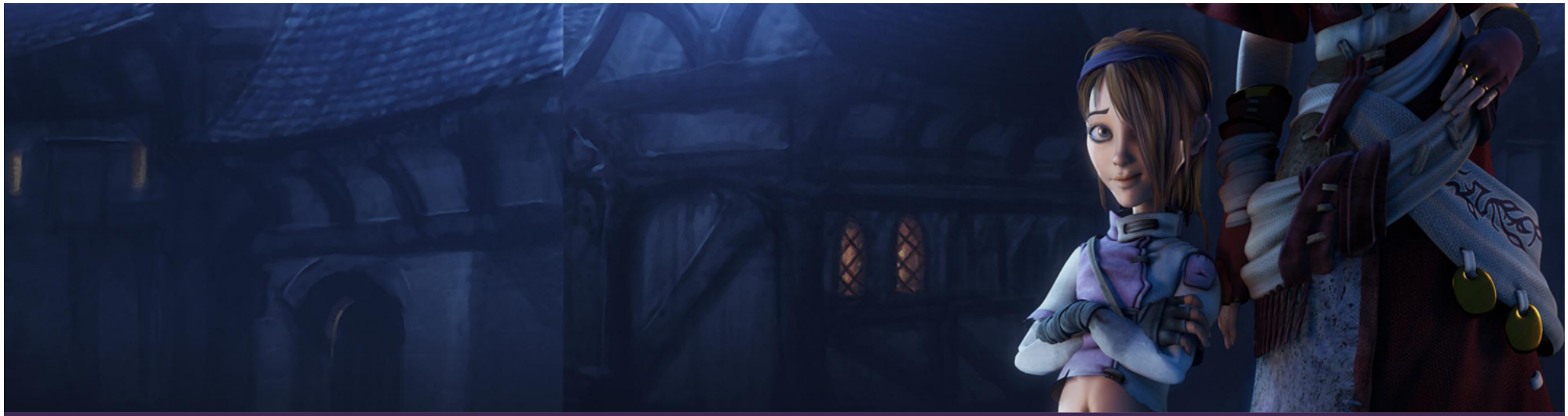


FALMOUTH
UNIVERSITY

Lecture 1: Introduction

COMP260: Distributed Systems
BSc (Hons) Computing for Games

- Today's session:
  - Introduction to the module
  - MUD and their pivotal role in the development of video games

- Introduction to the module

- Introduction to the module
  - A 12 week module to introduce you to distributed processing (networking)
  - Focus is on
    - Client / server software development
    - Threading
    - Managing remote servers
    - Games as a service

• Introduction to the module

Week 1	Week 2	Week 3	Week 4	Week 5	Reading Week
Intro & MUD	IP & Sockets	Threading	UX Programming	Peer Review	Studio Practice

Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
Assignment 1 Viva	Working with Databases	Client Accounts	Salting data	Encryption & Security	Peer Review

Week 13
Assignment 2 Viva

- Introduction to the module
  - Module is split into 2 parts:
    - Part I
      - Fundamental socket-based client/server application development
      - Python as a prototyping and development language
    - Part II
      - Software as a service development
      - Hosting and managing remote services (Digital Ocean services)
      - Python as a server language
      - C# as a client language
  - Assignments:
    - 2 development assignment
      - Build a multi-player MUD
      - Develop MUD as a service
    - Research journal on network security

- Introduction to the module
  - Delivery
    - Just like COMP220 in semester I
    - 1 hour lecture of theory
    - 1 hour session with Tris to develop your skills
    - 2 hour tutorial to explore the theory sessions
      - These will build into your assignments
    - NB:
      - This isn't the schedule for each week
        - » Peer review weeks will be different!

- Introduction to the module
  - Help, help
    - I don't know / I can't remember Python
      - https://www.pluralsight.com/courses/python-getting-started
    - I don't know PyQT
      - https://wiki.python.org/moin/PyQt
    - I don't know C#
      - https://app.pluralsight.com/library/courses/c-sharp-fundamentals-with-visual-studio-2015
    - Don't forget, you can pair program, form a self-help group / action learning set
      - https://rapidbi.com/action-learning-sets/

- MUD and their pivotal role in the development of video games

- MUDs
  - We are in the 1970s, first thing to consider is that this if **before** the home computer revolution of the early 1980s
    - So, no
      - smart phones
      - Laptops
      - Ipads / tablets
      - PCs / macs and so on
    - No internet as we know it
  - Early personal computers:
    - 1977 Apple II ($1300 ->$5k in today's money)
    - 1981 ZX81 (£100 -> £250)
    - 1981 BBC Model B (£350 -> £875)
    - 1982 C64 ($600 -> £1500)

- MUDs
  - University computer facilities were mainframes and minicomputers with dumb terminals

- All processing occurred in the mainframe, terminals just took input & displayed responses

- MUDs
  - University computer facilities were mainframes and minicomputers with dumb terminals
    - Early form of the MVC (model, view, controller) pattern
      - Mainframe had the ‘model’ as working memory or disk storage or relational database
      - Mainframe processed model with controller application (ASM, C, COBOL etc)
      - Dumb terminals allowed users to send commands to mainframe app and receive results as a view.
    - Just like modern cloud computing

- MUDs
  - Multi-user dungeons first became popular in academic environments in the mid-1970s
    - 1975 University of Illinois
    - 1978 Essex University ‘MUD’
  - Created by students rather than staff
    - Relied on university mini computers (DECs running Unix)
    - And academic networks (JANET, PLATO) to link players to MUDs
      - Often full of very sardonic humour based on campus experiences / staff / students

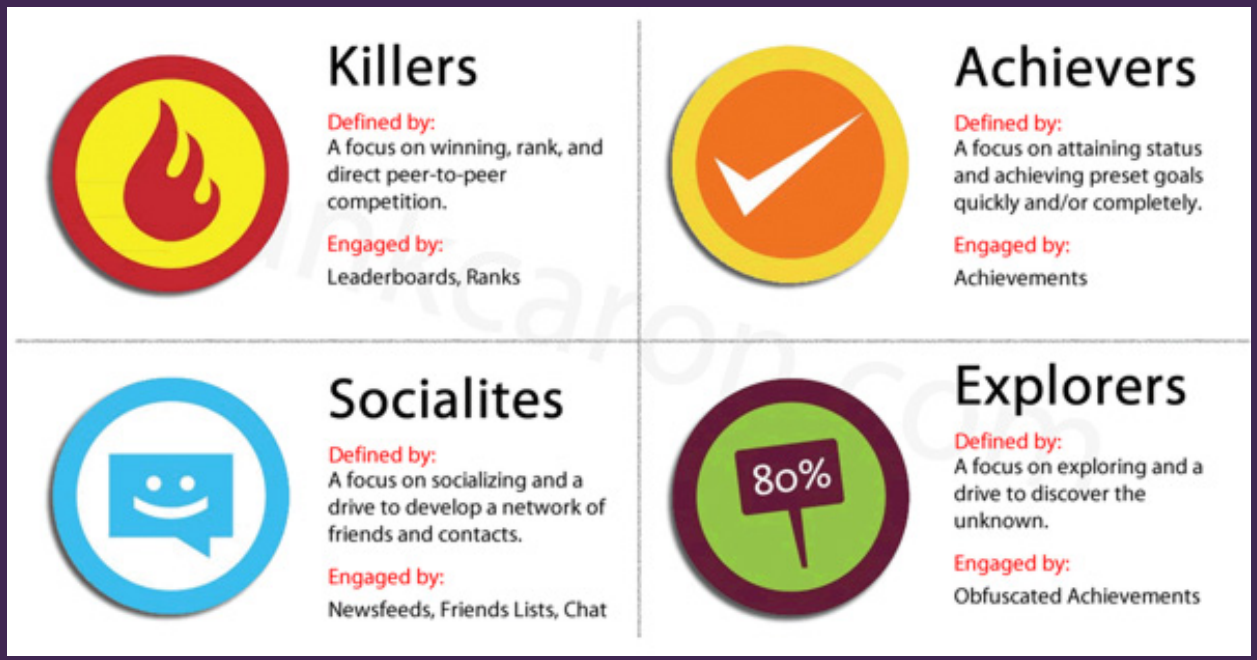
- MUDs
  - What is a MUD?
    - Simplest definition, it's digital 'Dungeons and Dragons' type role playing
    - Game consists of (simplest form):
      - A dungeon of inter-connected rooms
      - Human players
      - Text-based interactions

- MUDs
  - What is a MUD?

```
IDLE 1.1      ==== No Subprocess ====
>>>
Action: look
You Appear To Be At What Is The Start Of An Underground Dungeon
What Adventures Await You Below The Earth...
There Are Some Shiney Golden Coins On The Ground
There Is A Small Almost Transparent Bag In The Middle Of The Room
Action: pick up item
Select An Item To Pick Up
Item: bag
You Pick Up The Bag
Action: check item
Select An Item In Your Inventory To Examine
Item: bag
Light As A Feather, This Magical Bag Seems To Be Bottomless Allowing You To Store Any Item
Action: pick up coins
You Pick Up The Gold Coins
('You Now Have', 100, 'Gold Coins')
Action: check coins
('You Have', 100, 'Gold Coins')
Action: move forward
You Move Forwards To Room 1
Action: look
You Are In A Stone Cavern With A Dark Lake Stretching Into The Distance
Straight Ahead You See An Archway That Seems To Enter Somekind Of A Structure
There Are Some Shiney Golden Coins On The Ground
You See A Brightly Lit Torch On One Wall
Action:
```

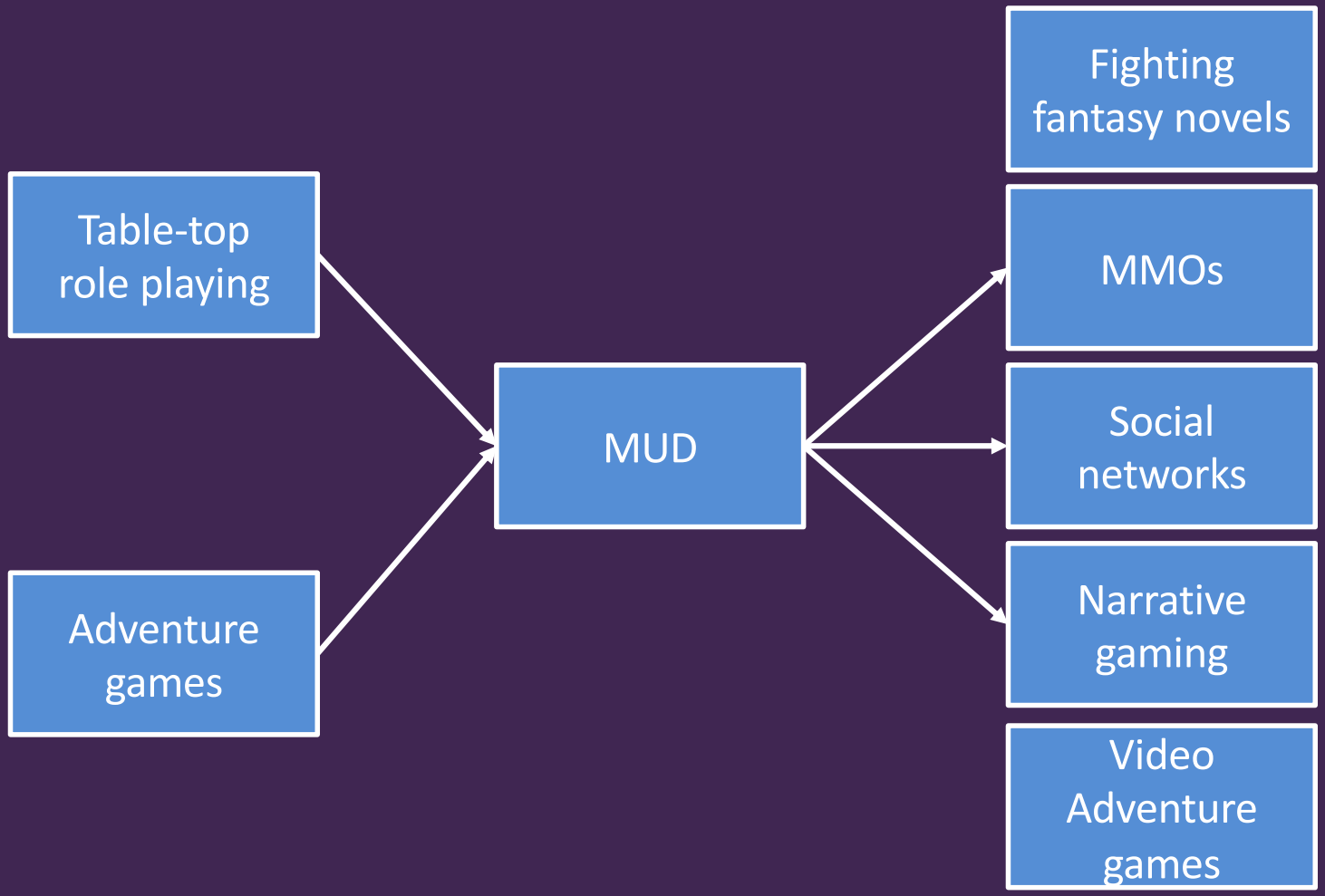
- MUDs
  - What is a MUD?
    - Simplest definition, it's digital 'Dungeons and Dragons' type role playing
    - Scope for:
      - A defined, connected and explorable world
      - Dungeon-based narrative / plot
      - Collectable game objects
      - Object/action challenges
      - NPCs to drive plot
      - Player interactions (chat / give objects / perform magic)
      - PvNPC combat
      - PvP combat
      - 2D/3D gfx

- MUDs
  - What is a MUD?
    - Bartle's player Types



https://repignite.com/2014/07/richard-bartle-player-types/

- MUDs
  - Why are they pivotal?



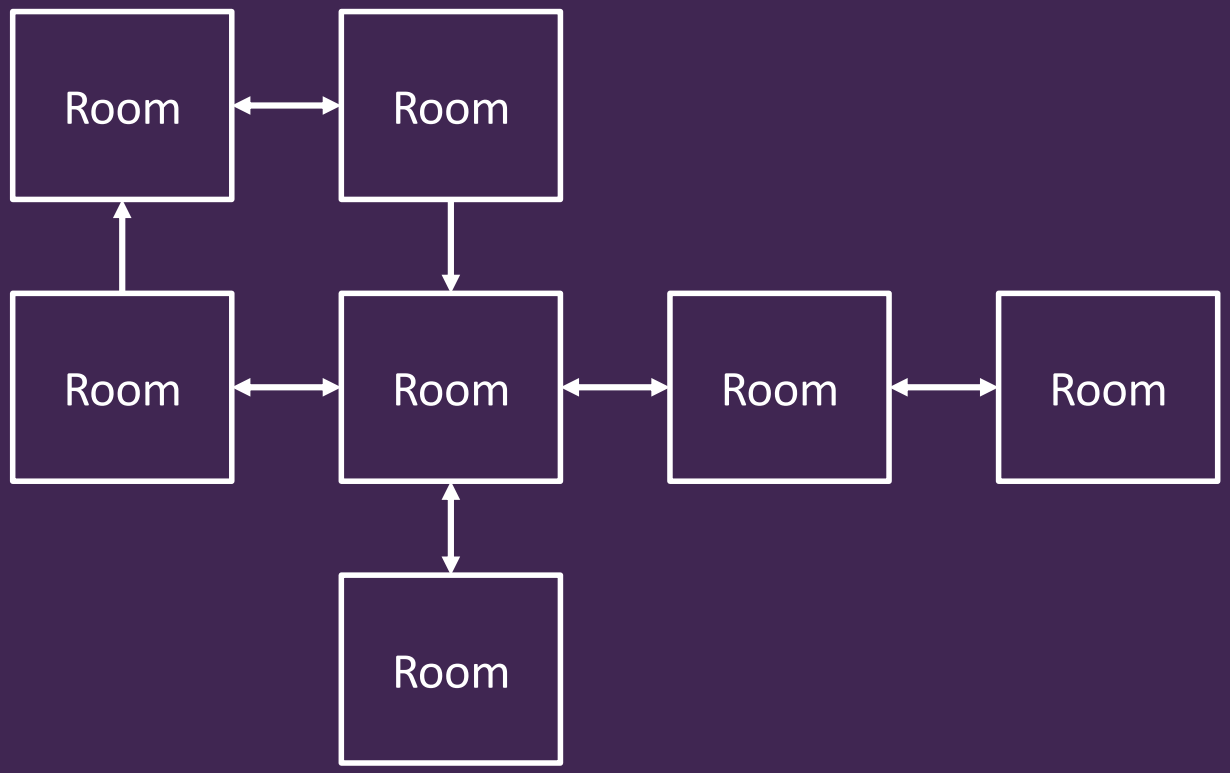
- Making a Single-user Dungeon

- Making a Single-user Dungeon
  - To make a SUD, we need to start off with some creative design that we can build our app from.

The SUD is a collection of rooms that are laid out on one level (there are no up or down exits). Rooms can be joined by north, south, east or west connections. A room does not have to have connections in all directions and it may not have more than one connection per direction and the connections do not have to be bidirectional.

When the player enters a room, they will be presented with a textual description of the room along with a list of exits they can take from that room. Typing help will give the player all of the text commands. Entering a command incorrectly will result in an error message.

- Making a Single-user Dungeon
  - To make a SUD, we need to start off with some creative design that we can build our app from.



- Making a Single-user Dungeon
  - Do a linguistic breakdown of the design,
    - Objects (nouns), methods (verbs) , attributes (adjectives)

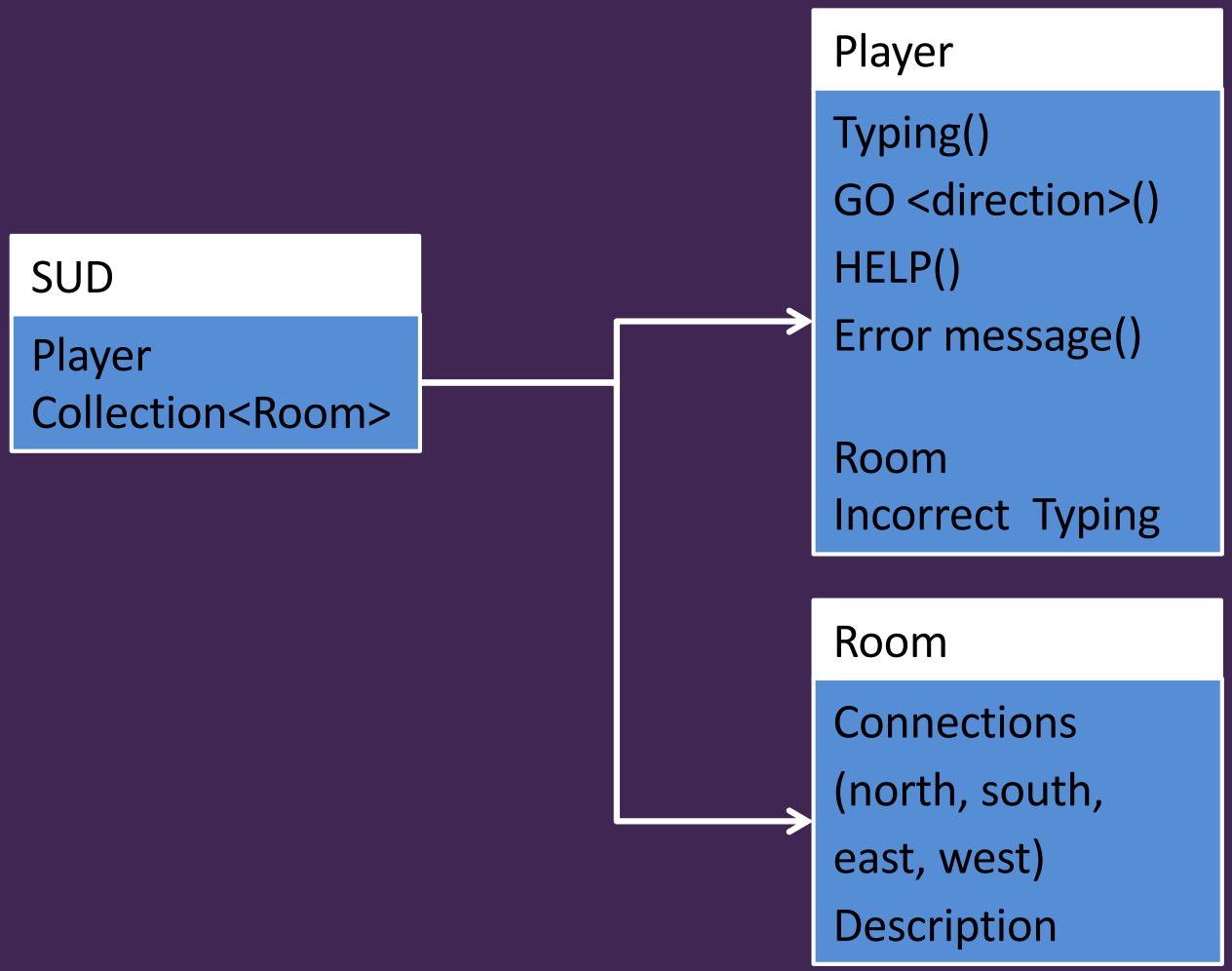
The SUD is a collection of rooms that are laid out on one level (there are no up or down exits). Rooms can be joined by north, south, east or west connections. A room does not have to have connections in all directions and it may not have more than one connection per direction and the connections do not have to be bidirectional.

When the player enters a room, they will be presented with a textual description of the room along with a list of exits they can take from that room. Typing help will give the player all of the text commands. Entering a command incorrectly will result in an error message.

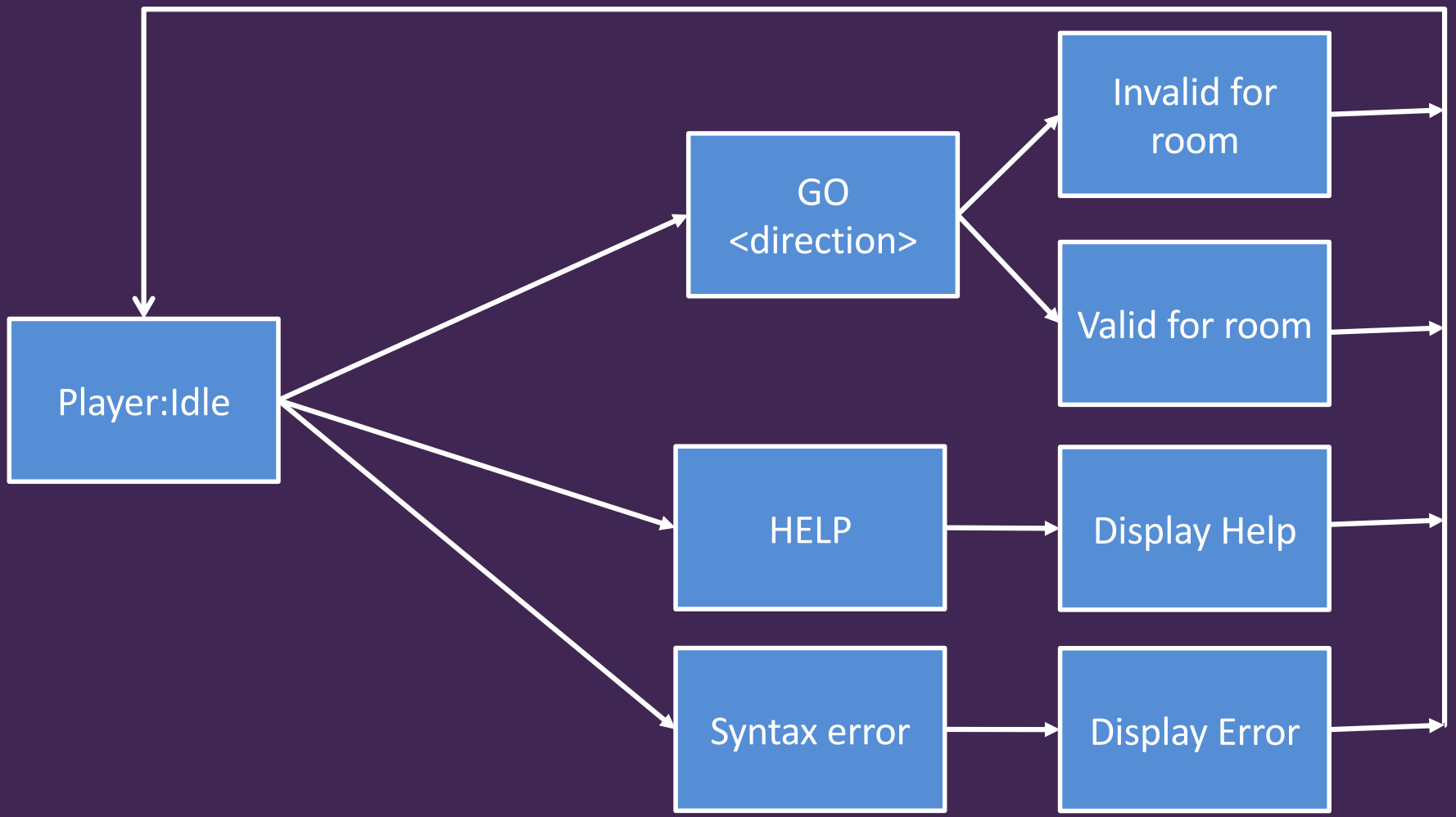
- Making a Single-user Dungeon
  - Create a 1st pass class hierarchy

Class	Method	Attributes
SUD		Player Collection<Room>
Player	Typing GO <direction> HELP Error message	Room Incorrect typing
Room		Connections (north, south, east, west) Description

- Making a Single-user Dungeon
  - Create a 1st pass class hierarchy



- Making a Single-user Dungeon
  - Create a 1st pass state diagram



- Making a Single-user Dungeon
  - What are the open issues to address
    1. How to store dungeon and room data
    2. How to make a text-driven interface
    3. How to process user input

- Making a Single-user Dungeon
  - 1.How to store dungeon and room data
    - From the class hierarchy:
      - The dungeon will be collection of rooms
      - Each room will contain a description and links to other rooms
    - What containers make sense for this?

- Making a Single-user Dungeon
  - 1.How to store dungeon and room data
    - From the class hierarchy:
      - The dungeon will be collection of rooms
      - Each room will contain a description and links to other rooms
    - What containers make sense for this?
      - Could use an array
        - » Does it make sense to index rooms from 0
          - Will this cause issues for our designers?
      - Could use a list
        - » Put room name into each room
        - » But that will give us an O(n/2) look up, do we want that?

- Making a Single-user Dungeon
  - 1.How to store dungeon and room data
    - From the class hierarchy:
      - The dungeon will be collection of rooms
      - Each room will contain a description and links to other rooms
    - What containers make sense for this?
      - Could use a dictionary of <string, room>
        - » Means we can use the name of the room as a key

- Making a Single-user Dungeon
  - 2.How to make a crappy text-driven interface
    - For prototyping, command line applications can be very useful
      - Just need to work out how to read text & display it
      - In Python:
        - » input()

- Making a Single-user Dungeon
  - 3.How to process user input
    - User input will be a string of characters
      - Maybe one word: HELP
      - Maybe two words: GO <direction>
    - Need to be able to process these strings to see what they contain and if they are valid commands
      - Split string using spaces into an array of strings
      - In python
        - » string.split(‘ ‘) # split string into an array using ‘ ‘ as a separator
          - Remember to remove any empty string from the array, if the user types lots of spaces

- Making a Single-user Dungeon
  - 3.How to process user input
    - Is input valid?
      - As a developer, you know what the command formats should be, so test accordingly
      - Converting input to lower case makes it easier to do comparisons ;)
      - In Python
        - » string.lower()

- Making a Single-user Dungeon
  - 3.How to process user input
    - Process input
      - In Python
        - » Use nested if/else to test for each command
        - » If all the ifs fail (because the user types rubbish)
          - Have a fall out case to tell the user their input is no good

- Worksheet 1: Make a Single-user Dungeon

- Questions