Simple Text Adventure Game (STAG)

COMSM0086

Dr Simon Lock & Dr Sion Hannuna

Database Marks & Feedback

Marks for DB are now visible in the system

Various feedback files can be found on Blackboard

Blackboard likes to create duplicate files (soz)

The four files to look at are:

```
feedback_ab12345_overview.txt
feedback_ab12345_failures.txt
feedback_ab12345_quality.txt
feedback_ab12345_inappropriate.txt
```

Mark Calculation

A "base mark" is calculated, based on tests passed
Penalties are then applied for quality and agility

Calculation of base mark uses a calibrated formula
To convert from "tests passed" to "percentage grade"

- Achieving 50% required 35 tests to be passed
- Achieving 70% required 48 tests to be passed
- etc.

We need to do this calibration step because:

- JUnit test cases don't respect class boundaries
- The level of challenge to students varies each year

Code Quality

We use automated checkers to flag potential issues
These are then manually investigated for severity
Some are "bad practice", others just "a bit strange"

Code quality feedback flags ALL identified issues
It is useful to be aware of both "strange" and "bad"

Worth mentioning that not all metrics are penalised
Some characteristics are more serious than others...

Important Quality Characteristics

Good and appropriate method and variable names

Method length, levels of nesting and complexity

Maintaining encapsulation (not using globals !)

Keeping your code DRY (avoiding copy-and-paste)

Factor out commonalities into reusable methods

Quality Issues Flagged Without Penalty

Comments are useful, but we don't enforce then

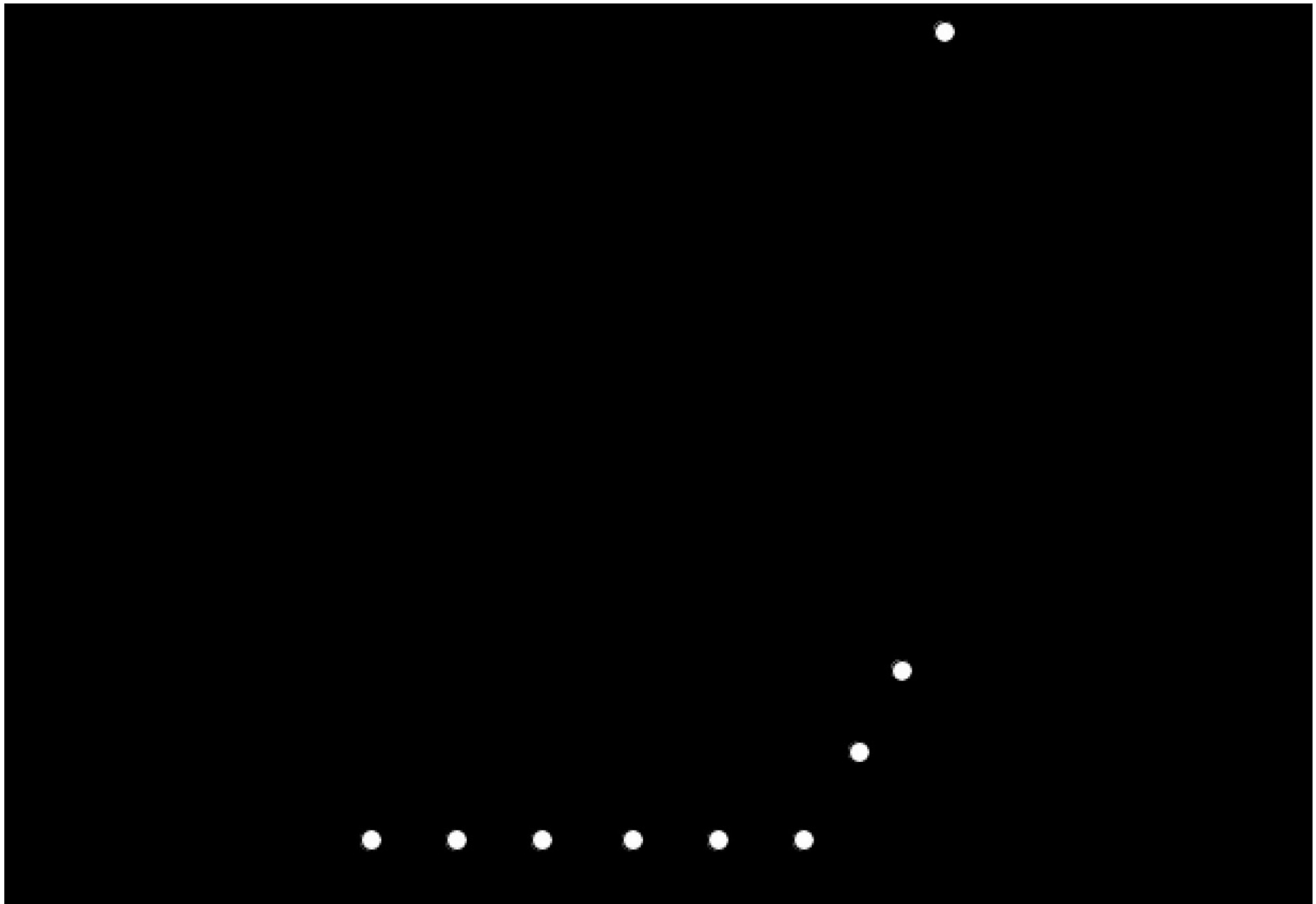
Blank lines help readability, but preferences vary

Indentation metric can sometimes be unreliable

Tight coupling is an important structural issue...

However some design patterns cause tight coupling !

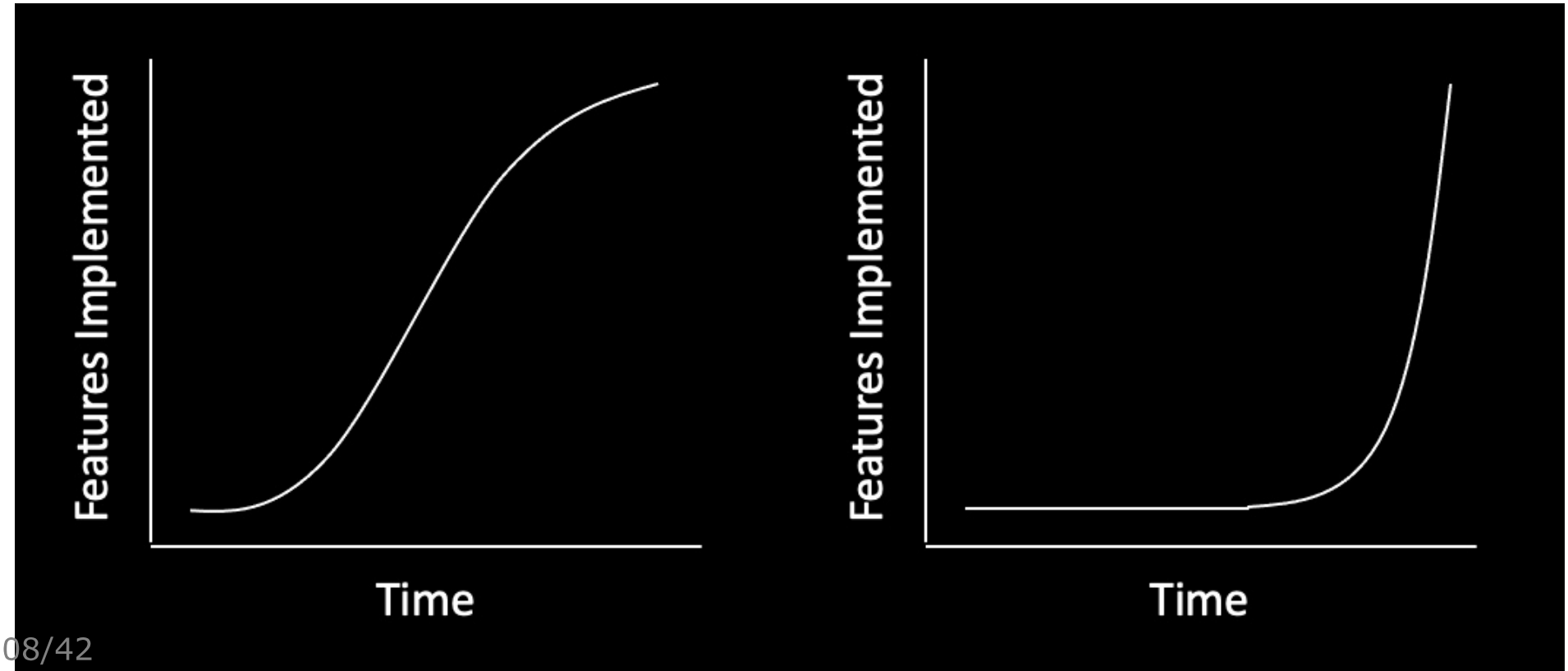
Agile "Steady & Sustainable" Feature Jump



Accumulation of Implemented Features

Your aim is to achieve the graph illustrated on the left

And avoid situation shown in the graph on the right !



Agile "Steady & Sustainable" Code Dump

18-03-24 100 lines of code committed

19-03-24 200 lines of code committed

20-03-24 600 lines of code committed

21-03-24 100 lines of code committed

22-03-24 200 lines of code committed

Commit Analysis

Can't just use raw commit data (can be misleading)
We need to filter the data in order to remove:

- files that aren’t .java source code
  (to ignore generated files on the repository)
- any java code inside test scripts
  (since these aren't part of the main codebase)
- lines moved from one place to another
  (since they aren’t new additions)
- lines merged in from another branch
  (these appear more than once in commit data)

Browse through your commit history for more details

Unexpected Commit Activity

Anticipated behaviour was 100 or 200 lines per day
Being pushed in a couple of commits during the day

Some repos show a massive amount of "thrashing"

- Many hundreds of lines being pushed in a day
- 1000 lines being merged from dev branch to main
- Dozens of commits in a single day

It all looks a bit chaotic and uncontrolled :o(

"Inappropriate" Files

In the overview feedback file you will see:

```
Penalty for pushing inappropriate file types to GitHub
Check your repo
```

A general warning about pushing inappropriate files

This was sent to all in order to encourage everyone to review the content of their repositories

Does not necessarily mean you received a penalty
See the "inappropriate" feedback file on Blackboard
This provides a log of inappropriate files (if any)

Extent of Penalties

Only minor penalties have been applied this time
Many people are still working in an un-agile fashion

In the next assignment penalties will be increased
You *should* know better by then !

Important that you develop in a mature fashion
We need to turn you into dependable team-players

Also, this kind of behaviour is masking plagiarism !

Now it's your turn !

We would like to get YOUR feedback
Tell us what you think about the unit
We are always looking to improve !

https://evaluation.bristol.ac.uk/home/

Simon's link:

bae4ea&sid=501556e3323e474e5e3059a7ba256c69a338b6

STAG

Overview

Final exercise is worth remaining 60% of assessment

The purpose of this assignment is to build...

A general-purpose socket-server game-engine
(for text adventure games)

If you are unfamiliar with the genre, take a look at:
https://tinyurl.com/zork-game

Game Server

The main class is a server listening on port 8888
This accepts incoming commands from clients
It should process command and change game state
Returning response to client (in required)

After processing command server closes connection
Then listen for the next connection on port 8888
Don't panic: this is provided for you in the template

Standard Gameplay Commands

- "inventory" (or "inv" for short): lists all of the artefacts currently being carried by the player
- "get": picks up a specified artefact from current location and adds it into player's inventory
- "drop": puts down an artefact from player's inventory and places it into the current location
- "goto": moves the player to a new location (if there is a path to that location)
- "look": describes entities in the current location and list paths to other locations

How can we play ANY game ?

Gameplay should be configurable by providing two "game description" files to the game engine:

- Entities: structural layout and relationships
- Actions: dynamic behaviours of the game

Before considering the content of these files
Let us discuss Entities and Actions at high level

Inheritance Hierarchy

Use a hierarchy of "Entity" classes in your game:

- Location: A room or place within the game
- Artefact: A physical "thing" within the game (that can be collected by the player)
- Furniture: A physical "thing", part of a location (that can NOT be collected by the player)
- Character: A creature/person involved in game
- Player: A special kind of character (the user !)

All entities will need a name and a description
Some sub-classes may need additional properties

The Location Class

"Location" is a complex entity which contains:

- Paths to other Locations (can be one-way !)
- Characters that are currently at a Location
- Artefacts that are currently present in a Location
- Furniture that belongs in a Location

Actions

Dynamic behaviour within game is represented by "Actions", each of which has following elements:

- A set of possible "trigger" key phrases
  (ANY of which can be used to initiate an action)
- A set of "subject" entities that must be available
  (ALL of which be present to perform the action)
- A set of "consumed" entities that are removed
  (ALL of which are "eaten up" by the action)
- A set of "produced" entities that are created
  (ALL of which are "generated" by the action)

Example Actions

The previous description of Actions
may be a little hard to comprehend !

Let's take a look at some examples to illustrate
Represented in a document language called XML...

actions

Entire Files

Rather than representing entities in XML
We will use an alternative language called DOT
(sorry about the name - not my choice !)

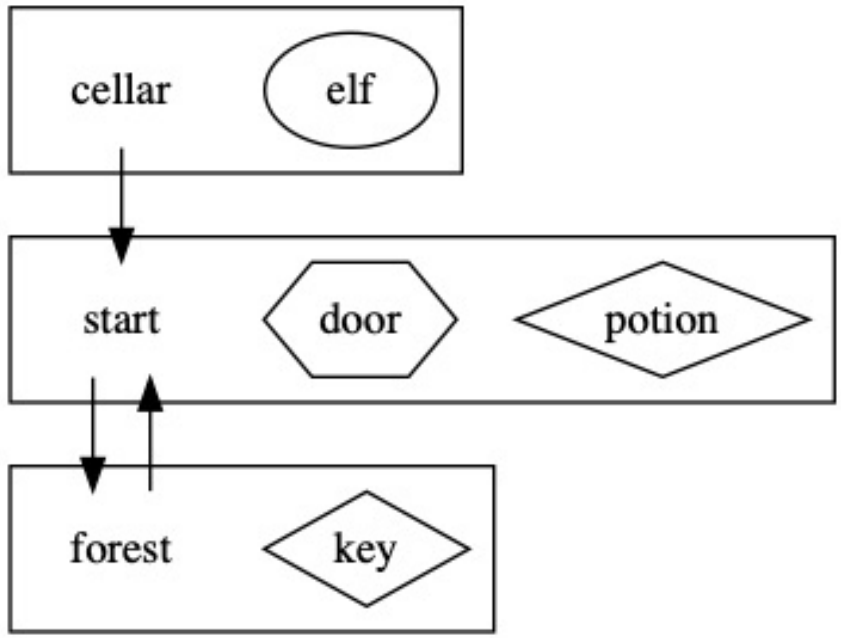
DOT is a language for expressing graphs
(which is basically what a text adventure game is !)

entities

Visualising the DOT files

The big bonus of using DOT files is that there are tools for visualising them (GraphViz !)

We can SEE the structure of the "entities" file:



Online Editor and Visualiser

20%20%20%20%20%20tree%20%5Bdescription%20%3D

Parsers

You've already experienced writing parsers
We don't want to cover old ground
So you'll be using two existing parsing libraries !

There is considerable educational value in learning to use existing libraries and frameworks

This can get lost in a desire to learn fundamentals
But not on this unit !

Which parsers ?

For parsing DOT you should use "JPGD":

http://www.alexander-merz.com/graphviz/doc.html

Library should already be embedded in maven project

And the Java API for XML Processing "JAXP":

http://oracle.com/java/technologies/jaxp-introduction.html

Core library which should be available to your project

Specifying Game Files

The two game configuration files will be passed as parameters into the GameServer constructor:

```
GameServer(File entitiesFile, File actionsFile)
```

This is essential so different games can be played
We have a special "marking" game configuration
So make sure your code can read in config files !

Unique Identifiers

Entity names in the configuration files are unique
So you can safely use these as unique identifiers

You won't have to deal with two things called "door"
There might be a "trapdoor" and a "frontdoor"

Think of them as variable names !

Action phrases are however NOT unique
(it might be possible to "open" many things)

Command Interpreter Flexibility

You interpreter needs to be able to cope with:

- Varying Case: All command are case insensitive
- Decorated Commands: extra "unnecessary" words
- Varying Word Order: triggers/subjects in any order
- Partial Commands: some subjects not mentioned

Full details for all of these contained in workbook

Correctness and Certainty

Your server should block any commands that contain extraneous entities

Stated by user, but not defined as subject of action

```
open door with key and axe
```

Your server should perform no action if command is ambiguous

Command matches more than one action

```
open with key
```

(if there are two doors, each with an "open" action)

Player Identification

Incoming commands begin with a username
(to identify which player has issued that command)

A typical incoming message might take the form of:

```
simon: open door with key
```

This allows the game to support multiple players !

Server does not need to deal with authentication

Health

As an extension to the basic game...
Add a "health level" feature to the Player class

Each player should start with a health level of 3
Poisons & Potions increase and decrease this number
(See the health related actions in the "Actions" file)

When player's health runs out, they return to start

Add a new "health" command keyword...
That reports back the player's current health level

Testing

A special set of custom game description files will be used to assess your game.
It is therefore essential your code is able to load files in the same format as examples provided !

Scripts will be used to automatically test your game engine to make sure it is operating correctly.
It is therefore essential that you adhere to the "built-in" commands detailed previously !

Code Quality

Code quality will again be assessed during marking

Adhere to guidelines outlined in the workbooks

Also review the quality feedback advice from DB !

(try to improve your bad habits !)

Agile

A key principle of Agile is providing value to client
Through *early* and *regular* delivery of features

Emphasis is on "Steady & Sustainable development"
(No "all-nighters", No "heroic effort")

Big chunks of work in short sessions is not Agile !
(And really upsets team mates and team leaders)

It is important you gain experience of Agile working
You'll be assessed on "Agileness" of your process

GitHub

In order to get insight into your dev. process...
You must push cw-stag to your existing GitHub repo

This will allow us to assess your WHOLE process
Not JUST marking the end product (the final code)

Investigate patterns of development work
Assess the accumulation of successful features

Identify occurrence of possible plagiarism
We have analysis tools to detect unusual behaviour

Your Process

You should commit and push to your repo regularly
After each coding session: "before you eat or sleep"

The master branch should *always* be operational
(that version of code should always be "runnable")

Your code has to be ready to run "out of the box"
(clone master branch and run server with maven)

What if you're working on version that doesn't run ?
Keep committing on a regular basis (track changes)
Only push commits to repo when code DOES run !

Plagiarism

Automated checkers used to flag possible plagiarism

If markers feel plagiarism may have taken place...

The incident is referred to faculty plagiarism panel

May result in a mark of zero for assignment

or even the entire unit (if it is a repeat offence)

Questions ?

Demo of Server in Action

RunGameServer

In order to connect to the server
We have also provided you with a GameClient:

RunGameClient

You won't need to alter the client code !