University of South Wales

# Coursework Introduction

Ben Daubney

# Coursework

University of South Wales

- Due date 27th April (6 term weeks, 5 teaching).
- Value - 50%
- We've covered the essentials of what you need.
- Will do custom controls next week – you might want to use this.

# Demo

University of South Wales

# Motivation

University of South Wales

- Content Creation Tools:
  - Many tools exist to create content for games, e.g. animate characters, create textures, sound etc.
  - Often companies will produce custom content creation tools specifically for their products/games.

# Problems

University of South Wales

1. XAML and C#.

2. Getting to grips with Ben's code.

   (you can do everything from scratch if you want!)

# Purpose of Coursework

University of South Wales

- Examine that you understand the concepts we have covered in lectures (more on this later).

- Get you used to working with other people's (which may not be as clean as you would like) code.

# Project Development

University of South Wales

If you did the same project twice would it take as long the second time?

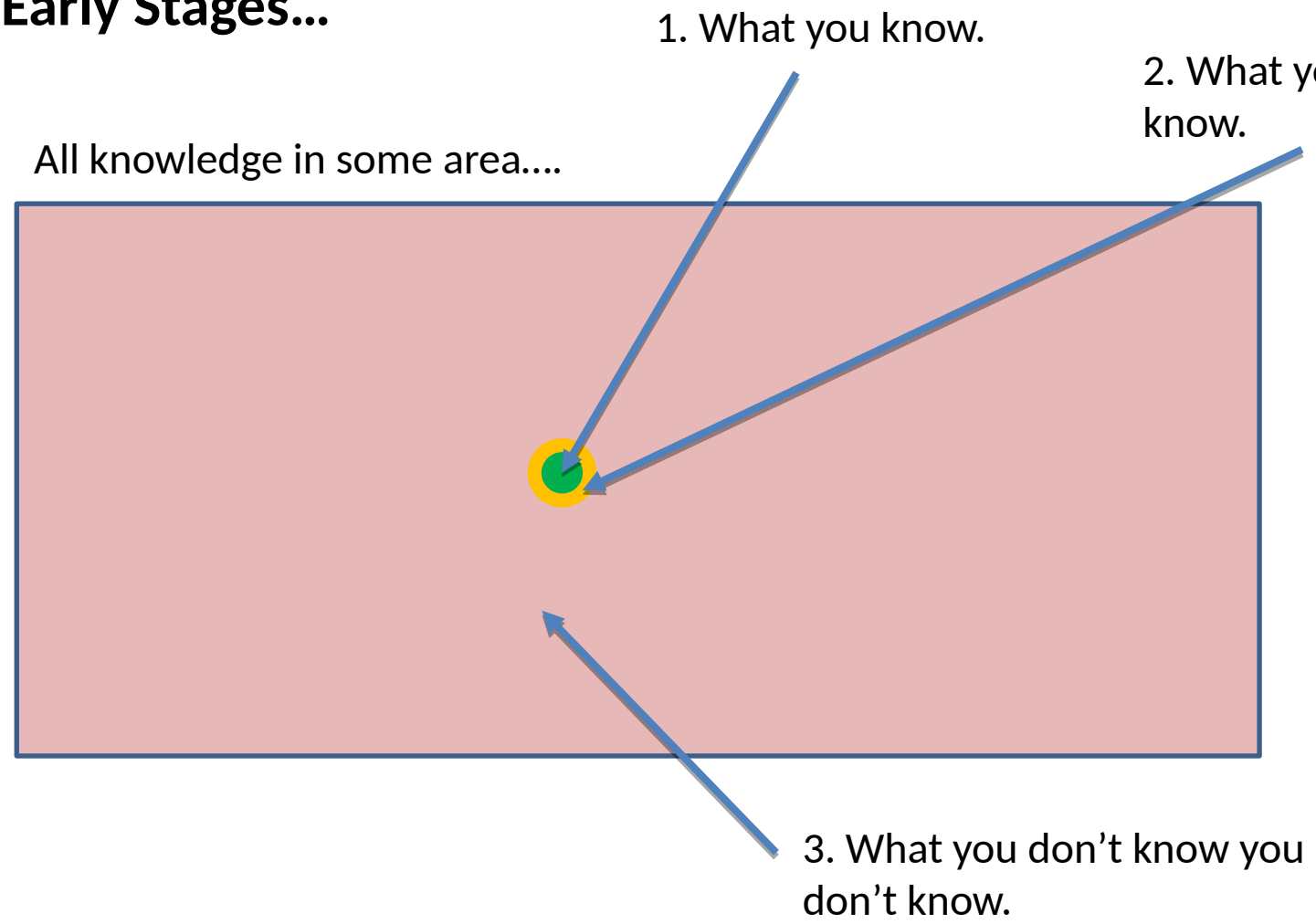
You have three levels of knowledge:

1. What you know.
2. What you know you don't know.
3. What you don't know you don't know (ignorance).

# Project Development

University of South Wales

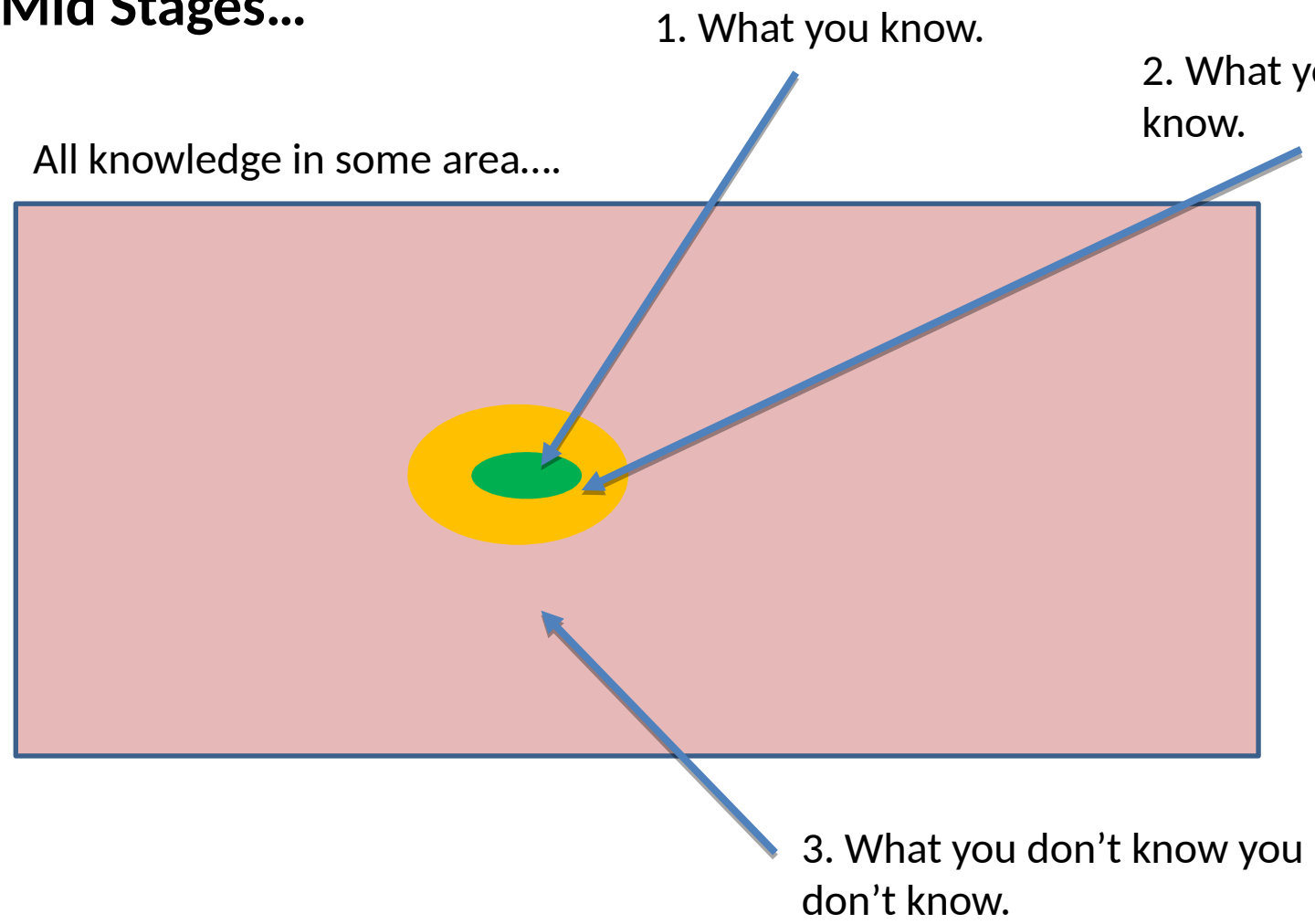
**Early Stages…**



# Project Development

University of South Wales

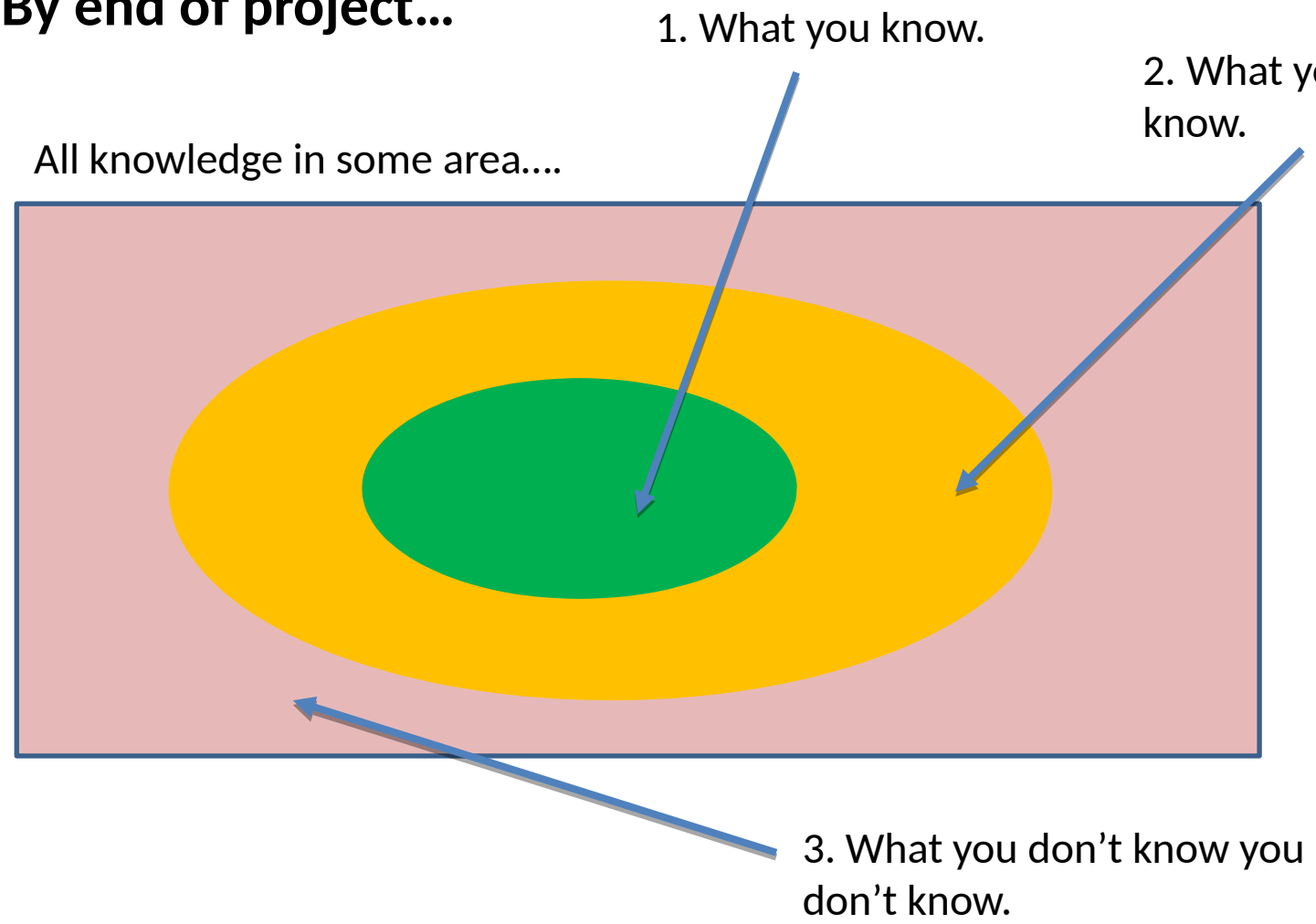
**Mid Stages…**



# Project Development

University of South Wales

**By end of project…**



# Project Development

University of South Wales

- By the end of a project you have more knowledge.
- However, you are more aware of how much you don’t understand.
- Can feel like you know less at the end than the beginning.

# Project Development

University of South Wales

- At the beginning of project you have least knowledge. This is the worst time to make big decisions about your design!

- Be prepared to make changes as you go along.

# Project Development

University of South Wales

- Decide whether you would rather tackle the knowns or unknowns first.
- In real world projects getting a handle on the unknowns allows risk to be managed.
- Coursework is different to real world projects, for coursework it’s good to get easy marks under your belt before you tackle more challenging aspects.

# Project Development

University of South Wales

- 80/20 rule:
  - It takes 20% of the time to solve 80% of the problem.
  - It takes the remaining 80% of the time to solve the remaining 20%.

# Project Development

University of South Wales

- For coursework 60/40 rule:
  - You can get 60% of the marks in 40% of the time.
- Therefore, first just try to get a solution working.
- Then if you have time, try to improve it make use of what we have covered in lectures.

# Project Development

University of South Wales

- Use an iterative approach.
- Get things working first and then worry about improvements/modifications.
- Use some sort of source control.

# Marking Guidelines

University of South Wales

- Code needs to be commented.
- Functions and function parameters should be documented (for event handlers just give overview).
- You can choose your coding conventions but must be consistent, choose something known (c-style, camel case etc etc).
- Make sure GUI items are appropriately named (abbreviation indicates control type).

# Comments and Documentation

University of South Wales

- Think about who you are writing for:
  - Class/function documentation is targeted at users of the class/function. They may not be able to view code.
  - Code commenting is targeted at other developers who may want to maintain/update your code.

# Class Documentation

University of South Wales

- Describe what class does.
- Provide indication of how class is typically used (may refer to class methods).

# Function Documentation

University of South Wales

- Describe what function does.
- Describes function parameters and they’re purpose, how they are used.
- Describes any assumptions made by the method (e.g. valid input data, data already loaded).
- May contain description of how function works if it’s important for user to know.
- Use auto format (C# press “///”)

# Commenting Code

University of South Wales

- Purpose of commenting code?
  - Make code easier/faster to understand?
  - Allow author to indicate intent.
  - Allow author to indicate their thought process.

# Bad Comments

University of South Wales

- Explicitly says what the line of code does:

```
// Close the client.
client.Close();
```

- Is incorrect:

```
// Open the client.
client.Close();
```

- Is irrelevant/mildly offensive:

```
// You work it out sucker.
client.Close();
```

# Commenting Code

University of South Wales

- Sometimes better to comment blocks of code, rather than individual lines.

```
////////////////////////////////////////////////////////////
// register callback and start listening for messages.

receiver.OnUpdate += new UpdateCallback(UpdateSummary);
receiver.OnUpdate += new UpdateCallback(UpdateTable);

receiverThread = new Thread(new ThreadStart(receiver.DoSomething));
receiverThread.Start();


////////////////////////////////////////////////////////////
// Do next job…..
```

# Good Coding Practices

University of South Wales

- Declare variables when you need them.
- Don’t give variables scope beyond when you need them.
- Don’t try to optimise code.
  - If code is clearer in 12 lines than 1 leave it as 12.
- Remove unused variables/function parameters.
- Consider whether all class members/methods are necessary. (function size??)
- Ensure Private/Public/Protected is correct.

# Code Quality

University of South Wales

- Re-useable.
  - Try to limit code duplication (event handlers).
- Understandable.
- Maintainable.

# Reusing my Code

University of South Wales

- You don’t lose marks for re-using given code.
  - You don’t need to re-comment any given code.
  - You may want to refactor sections if it proves useful to you/makes code more reusable.

- However, the XAML section on the GUI interface is poor (thus this is the focus of the coursework).

# Reusing my Code

University of South Wales

- Don’t spend ages trying to fully understand the given code.

- You will understand the important bits as you progress.

# Leaving the Game In

University of South Wales

- I would prefer you to remove the parts of code that make the game work (e.g. so you can't play the game through the CW you hand in).

- If you don't remove the game parts you won't lose marks but make sure the game always works even after editing code.

# What have we covered?

University of South Wales

1. A variety of basic controls.
2. How to handle events on a specific control.
3. How to make use of the sender object in a event handler.
4. How to make use of the event args (e.g. mouse position).
5. How to set an event handler on a panel to handle child events.
6. Using panels for different layouts.
7. Adding panels as content.

# Marking Scheme

University of South Wales

| | |
|---|---|
| The tool must be able to load and display a level in its initial configuration. | 10 |
| The tool must allow a user to toggle a tile between floor and wall. | 20 |
| The tool must allow the user to change the bmp's used to represent floor and wall tiles, the knights, ghosts, fire and stairs. The appearance of the board should be appropriately updated. | 15 |
| The tool must allow the user to change the number of rows and columns contained in the tiled games, whilst maintaining existing data. Any items outside of the board should be moved to (0,0). | 20 |
| The tool must allow the user to be able to add and remove ghosts from the level by directly clicking on a tile. | 15 |
| The tool must allow the user to be able to define the start position of the knight and the location of the stairs. | 10 |
| The tool must allow the user to be able to save new levels that can be loaded into the game. | 10 |

# Marking Scheme

University of South Wales

- The purpose of your coursework is to check that you understand the material delivered in the course.

- Most of the marks are for demonstrating understanding (through code, comments and documentation).

# Advice

University of South Wales

- The layout I have given as an example in the coursework is not necessarily the best layout.
- Try to vary your controls/methods (think about usability).
- Try to demonstrate your knowledge!

# Advice

University of South Wales

- If you can't do something, partial solution is fine.
  - E.g.

    Can't get floor/wall to toggle. However, you did manage to get position of click and update level, just struggled with re-render. Will probably still get good marks, leave a comment as to where you got stuck.

- If you want to add a document explaining anything I need to know (e.g. bugs, how your program works) you can do and I will read it.

# Asking for help

University of South Wales

- I'm happy to help*
- I can't write code for you.
- I can explain examples/slides/tutorial solutions.

* Help provided proportional to perceived effort.

# Feedback

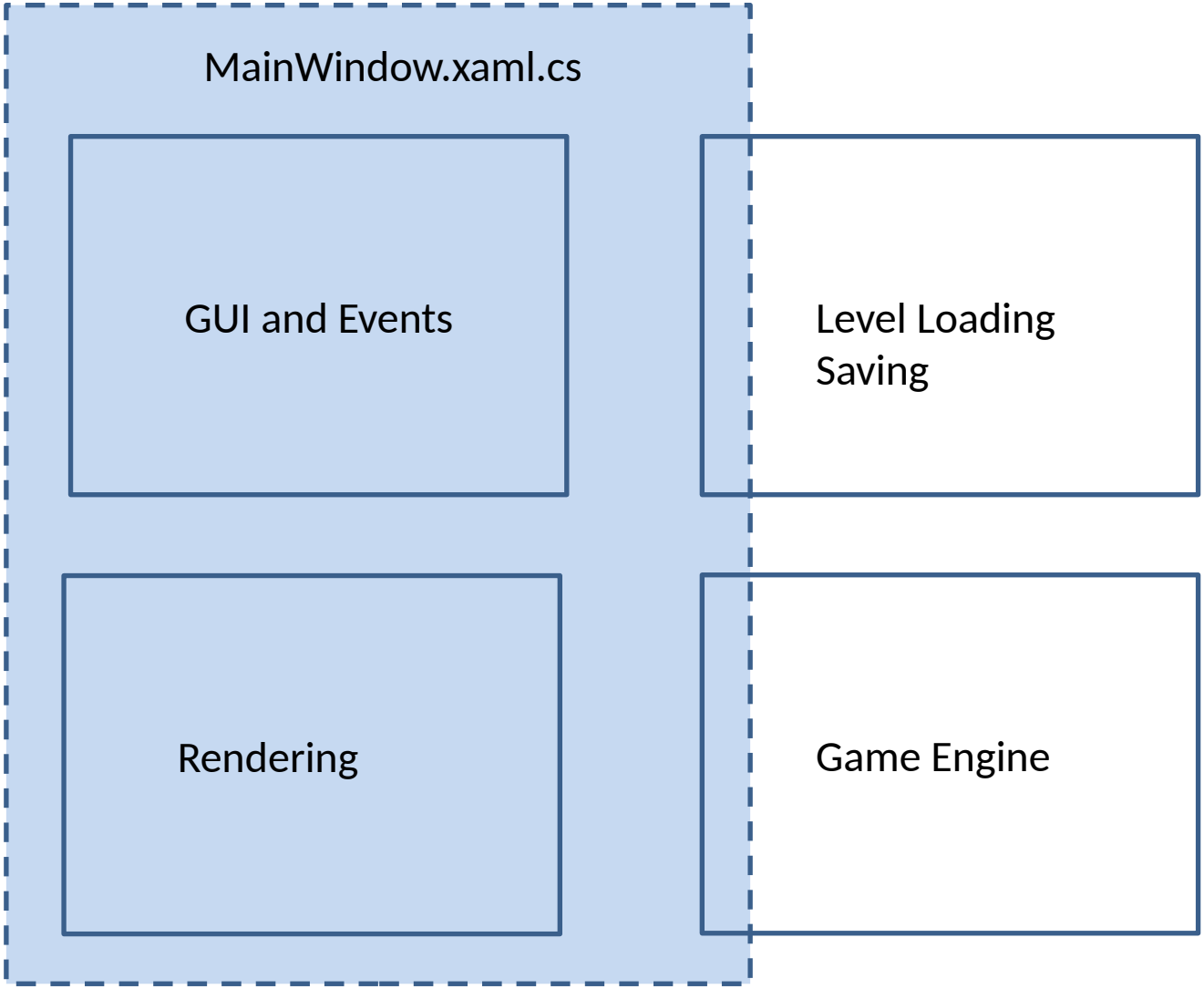
University of South Wales

- I will perform code review as well as overall comments.

- Just because I make a suggestion in the code review, doesn’t mean it lost you marks.

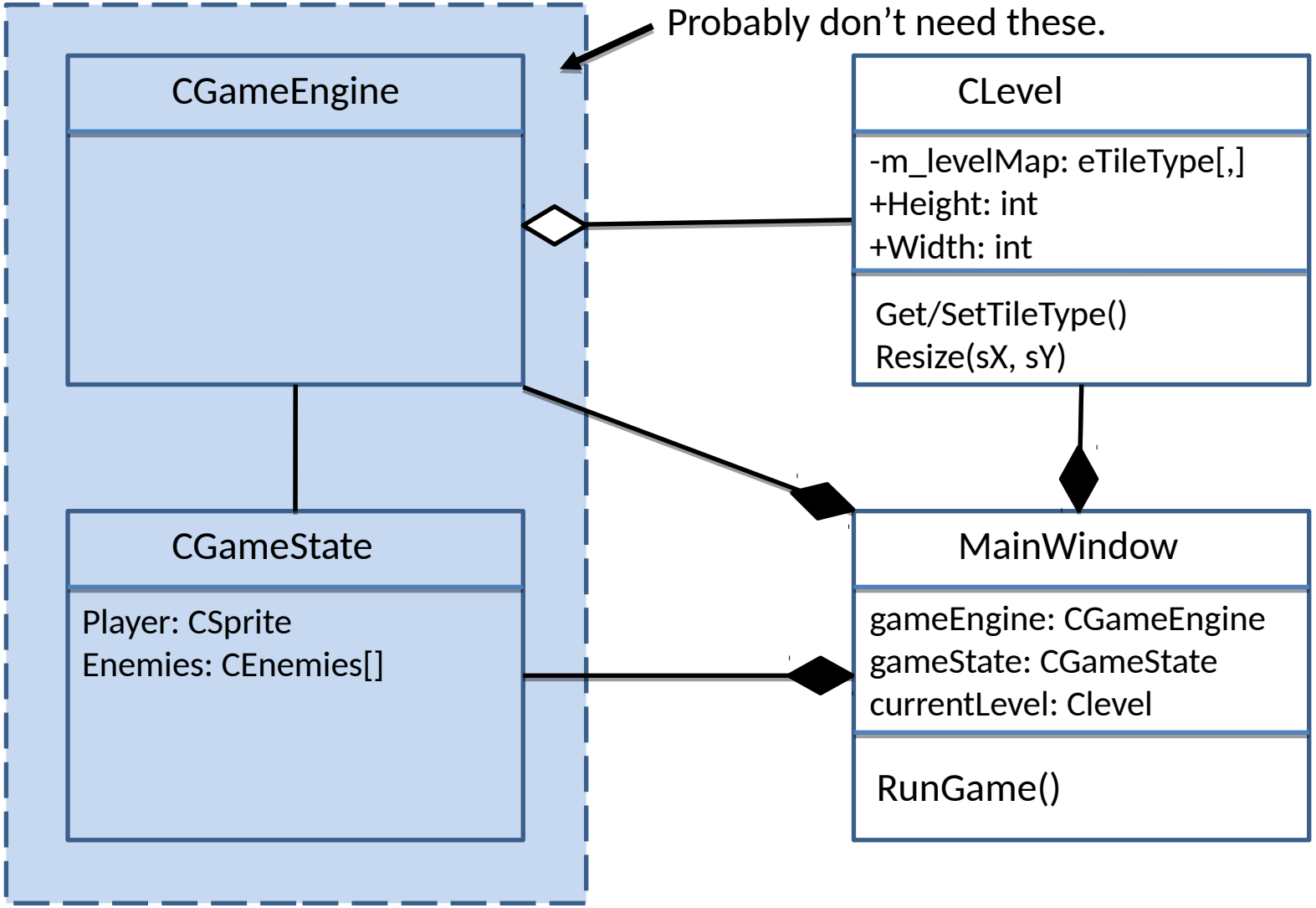
# Overview of Game

University of South Wales



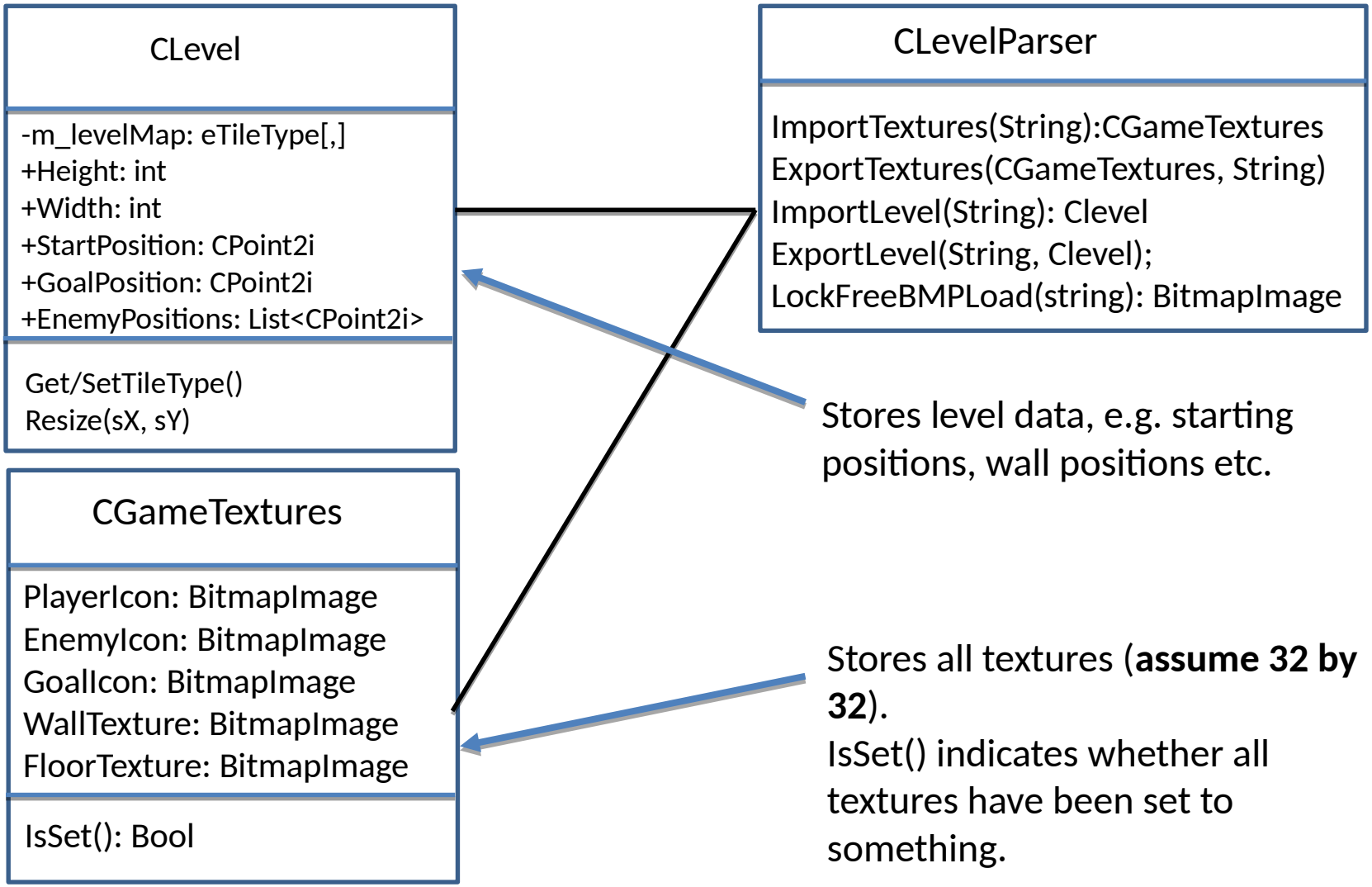
# GameEngine

University of South Wales



# Level Loading Saving

University of South Wales



# Level Loading Saving

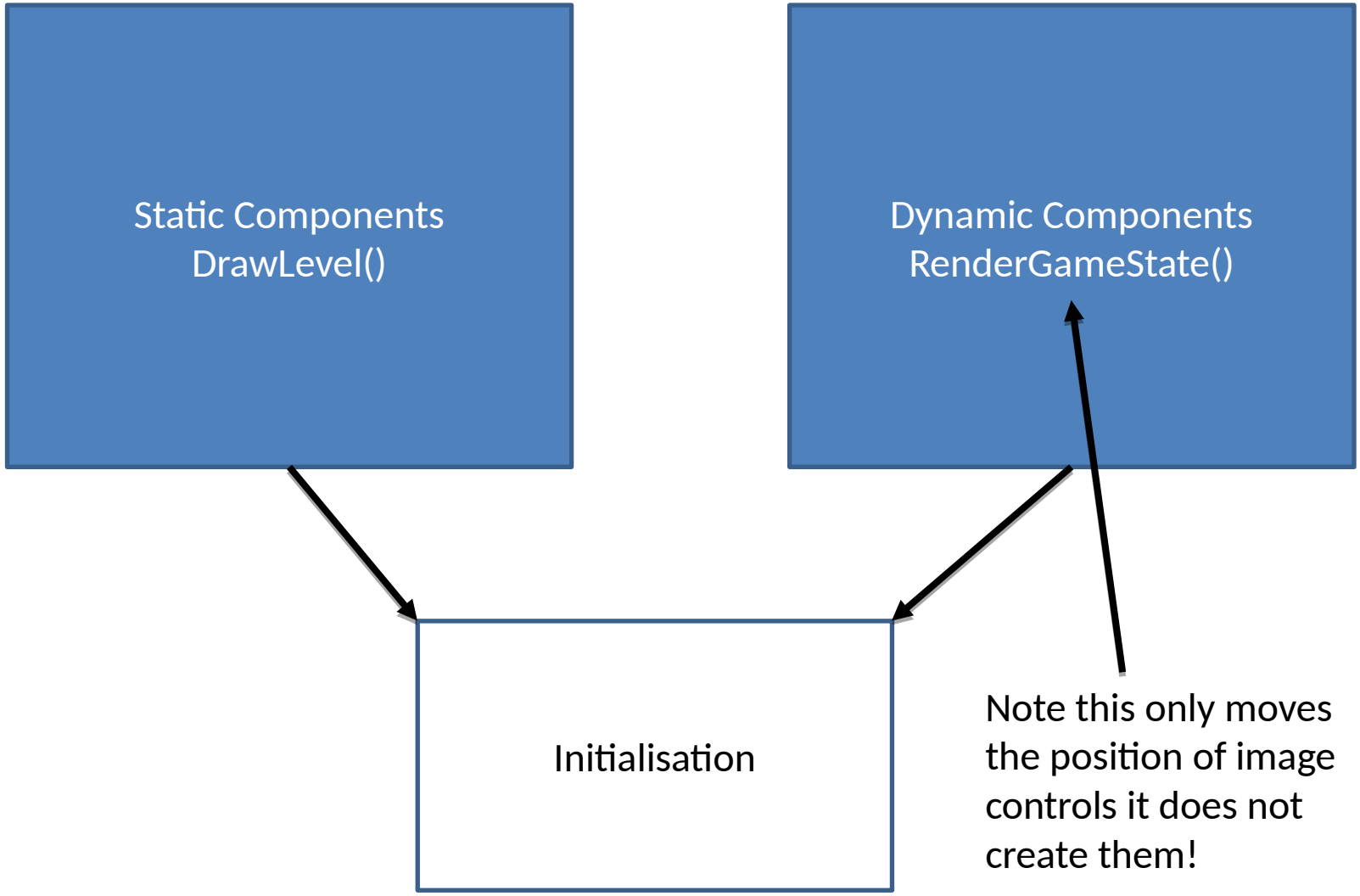
University of South Wales

*Helpful hint:*

**Rewriting the Level Parsers would be a very unnecessary thing to do!**

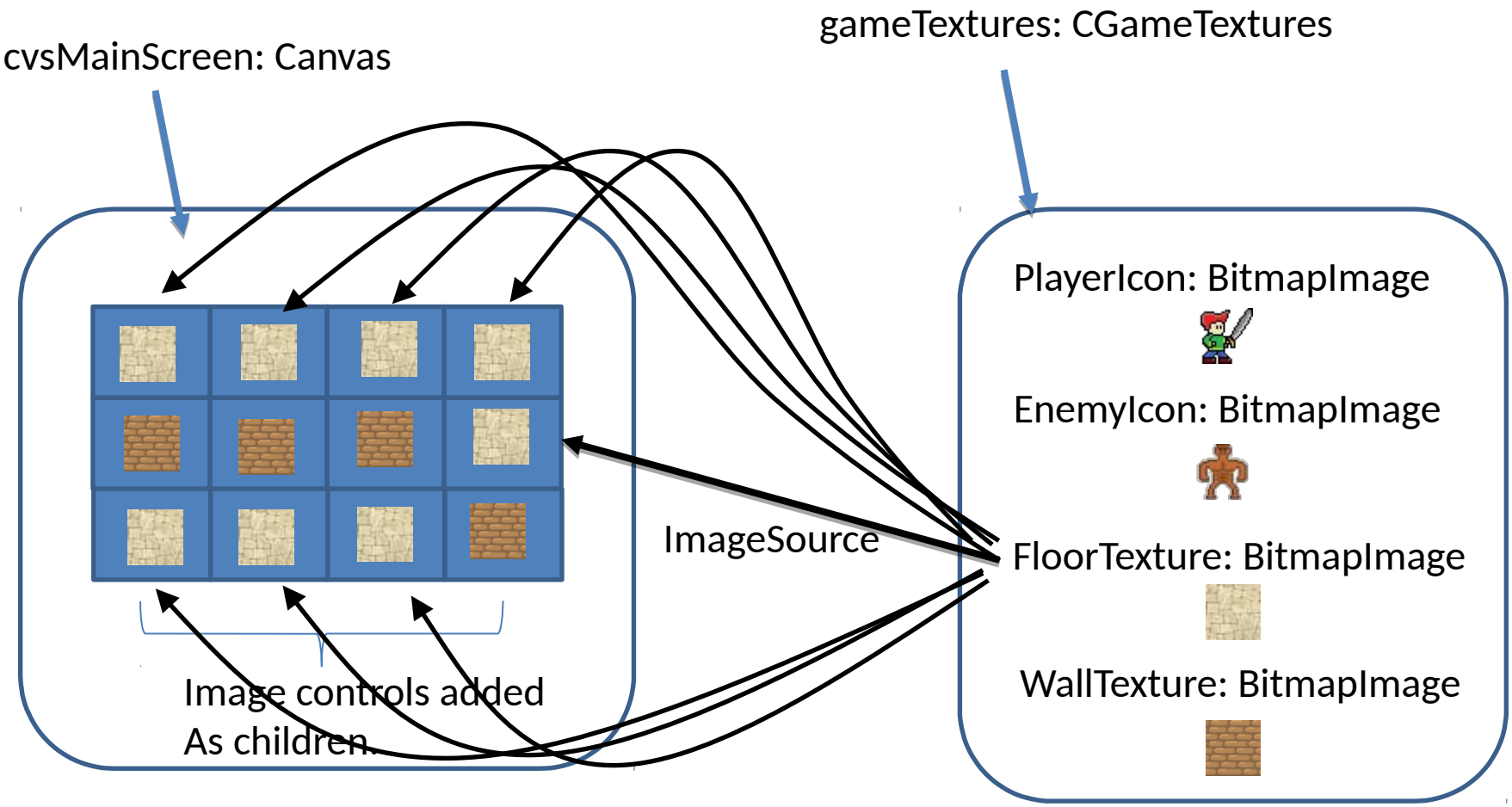
# Rendering

University of South Wales



# Rendering

University of South Wales



# Dynamic Scene Rendering

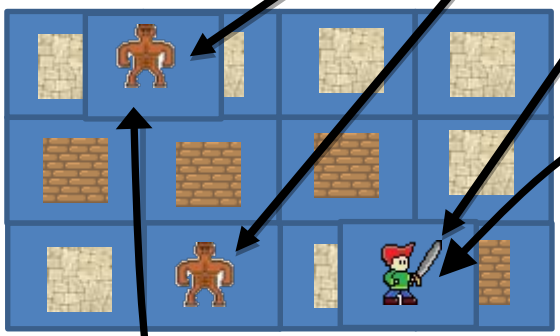
University of South Wales

We maintain a ref to Image controls to adjust their positions.

```
Image[]
enemyIcons;
Image
playerIcon;
```

cvsMainScreen: Canvas

gameTextures: CGameTextures



PlayerIcon: BitmapImage

EnemyIcon: BitmapImage

FloorTexture: BitmapImage

WallTexture: BitmapImage

Also added as control.

# Rendering

University of South Wales

- Will changing a BitmapImage on gameTexture automatically update all Image controls (tiles) that use that image?

No, the ImageSource would still reference the old image, the ImageSource must be refreshed on each control.

# Initialisation

University of South Wales

1. Clear all children from panel.
2. Load level and textures.
3. DrawLevel() -> Draws static parts of level.
4. Add Image controls for Player and Enemies (stored in array).
5. Set ImageSource for each Image control to relevant texture from gameTextures.
6. Create and initialise GameState from currentLevel.
7. RenderGameState.

Editor won't need these steps, can just use settings from currentLevel.

# Approaches

University of South Wales

- Changing a tile's appearance, texture:
  - Sledgehammer approach:

    Clear everything and redraw from scratch.
  - Tack hammer approach:

    1. Change the image source on only those Images that have changed.

    2. Change the position of only those Images that have moved.

# Other items of interest

University of South Wales

- CLevelUtils contains two static methods for converting between pixel coordinates and tile coordinates:
  - GetTileCoordinatesFromPixel(…)
  - GetPixelFromTileCoordinates(…)
- Usage:

```
CPoint2i PixelPosition = CLevelUtils.GetPixelFromTileCoordinates(TilePosition);
```

# Assumptions

University of South Wales

- Assume textures will be 32 by 32 pixels.
- Assume level directory for saving will be specified in text box (unless you can do otherwise).
- Assume a user may save to the same directory they loaded from.
- If you provide a text box for the user to enter number of rows/columns assume input is a number. I won't try to enter text (however, I may enter stupid numbers!).

  p.s…. In case you forgot -> int i= Int.Parse(txtMyNumber);

String containing number e.g. "45"

# A quick aside on Lists

University of South Wales

- To add an item:

  myList.Add(object);

- To remove an item:

  myList.Remove(object);

Removes by reference.

- To access number of items:

  myList.Count();

- To access item:

  myList[i];

# Questions

University of South Wales

Good Luck!