

Bulls & Cows Console Game Slides

[<< Back To Section 1](#)

[Go To Section 3 >>](#)

These are the slides that accompany the [Complete Unreal Developer Course](#).

See me develop the slides as I write the course...

- Right click or Insert > Comment to comment, especially if you see a typo
- A PDF version will be attached inside the Unreal course.
- The slides will update immediately as I change things.

Enjoy your stay!

Ben Tristem



View and comment online at <http://bit.ly/UnrealSlides>

Intro, Notes & Assets

@UnrealCourse :: www.UnrealCourse.com



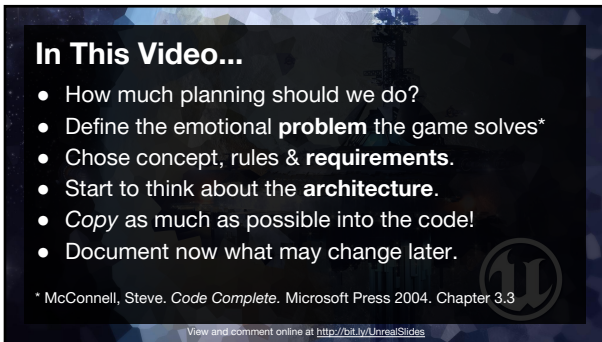
In This Video...

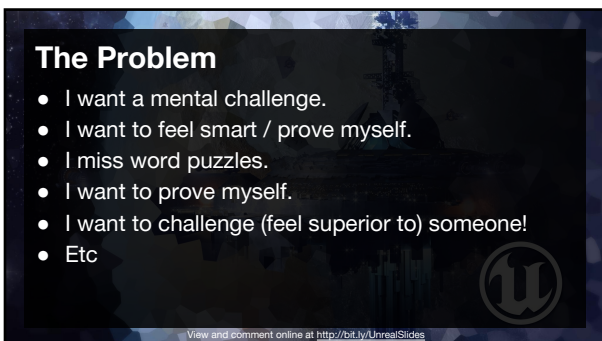
- Welcome to the first actual coding video.
- Why we're doing this in the IDE only.
- What you'll be building, see resources.
- You'll learn types, loops, routines, classes.
- We'll follow Unreal's coding style, and re-use.
- Notes and resources are attached.



View and comment online at <http://bit.ly/UnrealSlides>







Concept & Rules

- This is a “guess the isogram” game.
- An isogram is a word with no repeating letters.
- The user has a limited number of guesses.
- After each guess the computer outputs...
 - Bull = right letter in the right place.
 - Cow = right letter in the wrong place.
- You win by guessing the word within max tries.

View and comment online at <http://bit.ly/UnrealSlides>

Write Up The Requirements

- What will the **inputs** be? In what format?
- What will the **outputs** be?
- What **tasks** will the user be asked to do?
- Any **performance** limits worth mentioning?
- What **assets** (art, sound, story text) do we need?



Requirements

- Plain text instructions for all interactions.
- Code to help the player make a valid guess (e.g. all lowercase, an isogram, right length).
- Code to check the number of Bulls and Cows in the guess, compared the hidden word.
- Code to keep track of the number of valid guesses.

View and comment online at <http://bit.ly/UnrealSlides>

Possible Future Ideas (The NO List)

- Give feedback on every key press.
- Have a large dictionary of hidden words.
- User selectable word length, and difficulty.
- Checking the user's guess is a dictionary isogram.
- Providing a time limit for the guesses.
- A hint system, spend a turn for a hint.



View and comment online at <http://bit.ly/UnrealSlides>

How Solutions & Projects Relate

@UnrealCourse :: www.UnrealCourse.com

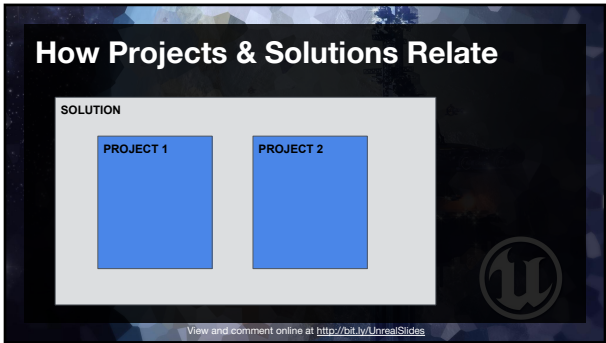


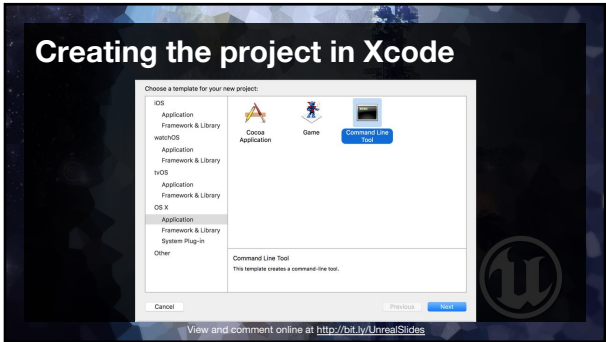
In This Video...

- How projects and solutions relate.
- Setting up a new command line project.
- An overview of the structure of our solution.
- (Adding `main.cpp` to our project).



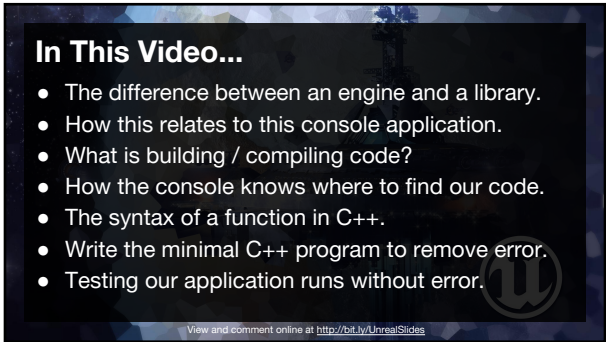
View and comment online at <http://bit.ly/UnrealSlides>

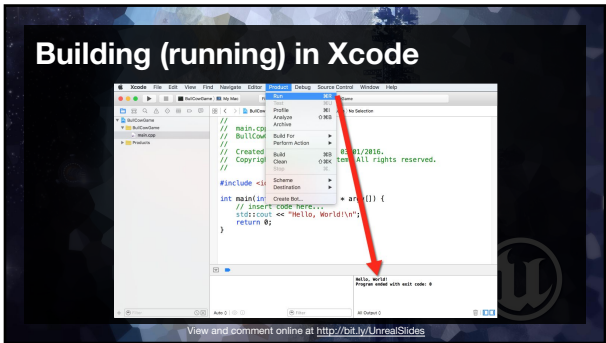












The syntax of a function in C++

```
int DoubleMe(int number)
{
    return number*2;
}
```

```
<return_type> <name> (<parameters>)
{
    <statements>
}
```

View and comment online at <http://bit.ly/UnrealSlides>



Write the minimal C++ program

- Return type is **int** (short for integer).
- Function name is **main** (lowercase m).
- Takes no parameters.
- Extra credit: make it return 0.
- Test by running and see if the error goes away.



Using, **#include** and Namespaces

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- `#` represents a “preprocessor directive”.
- `#include` copies-and-pastes other code.
- The idea of using library code.
- Use `<>` for standard libraries.
- Use `" "` for files you have created yourself.
- Notice the namespace icon in autocomplete.
- Import `iostream` library and use `std` namespace.

View and comment online at <http://bit.ly/UnrealSlides>

Using `cout` vs `printf()`

- There are pros and cons.
- You'll see both in other people's code.
- Read more at the link below.

<http://stackoverflow.com/questions/2872543/printf-vs-cout-in-c>

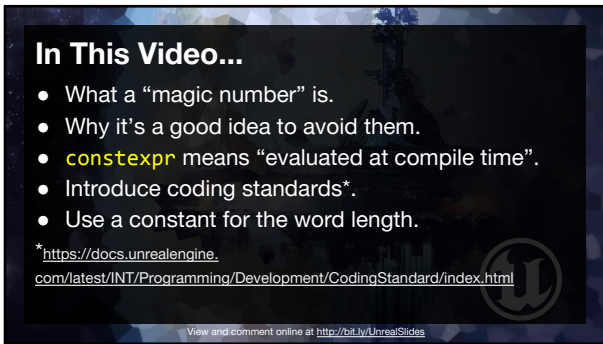
View and comment online at <http://bit.ly/UnrealSlides>

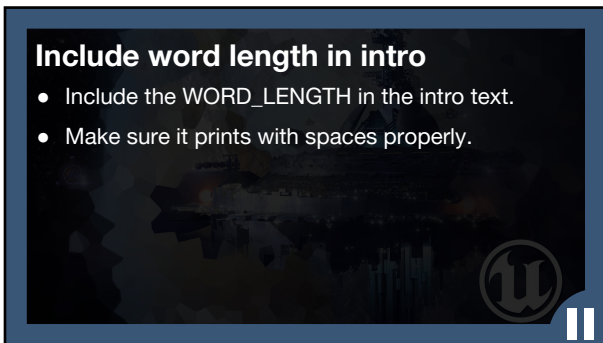
Use the `std` namespace

- Make appropriate use of the using statement.
- Test by removing `std::` prefix from your `cout`.
- Explain the risk in the discussions.

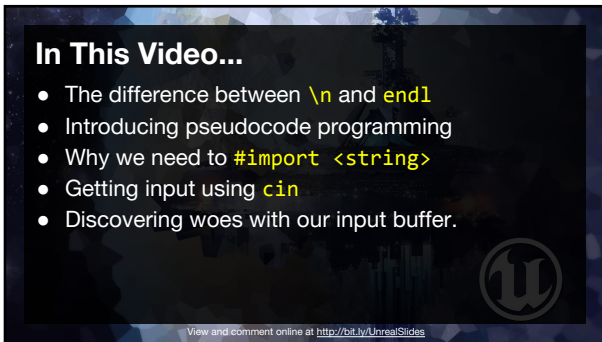


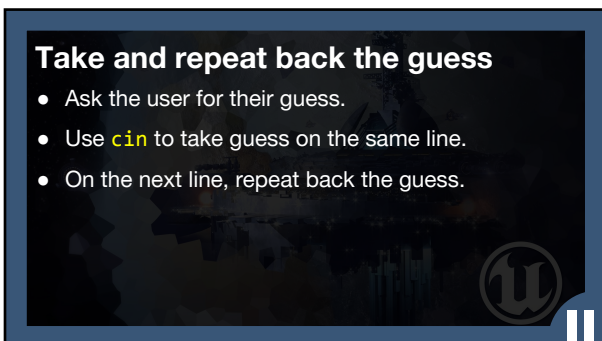




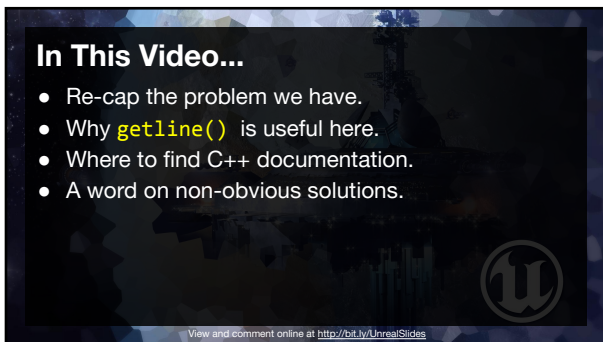


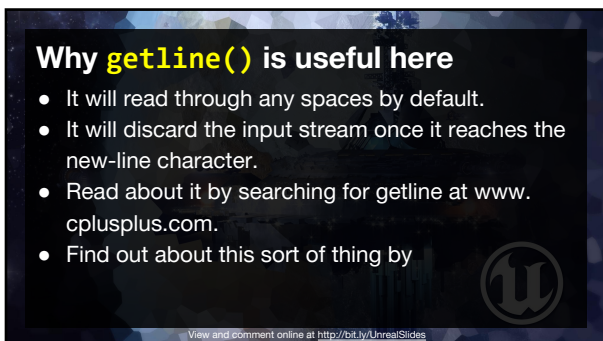












Fix the input problem

- Replace both **cin** lines.
- Test that donkey kong is accepted as a guess.



Simplifying With Functions

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- Programming is all about managing complexity.
- We want to think about a few things at a time.
- The idea of abstraction and encapsulation.
- How functions help us simplify.
- Write and call your first functions.
- A warning about “side-effects” of functions.
- Always use **return** at the end of your functions.

View and comment online at <http://bit.ly/UnrealSlides>



Abstraction and encapsulation

- A major goal in writing software is to manage complexity.
- **Abstraction** is a technique for managing complexity, by considering things at a higher level.
- **Encapsulation** is a way of making sure your abstractions are adhered to.



View and comment online at <http://bit.ly/UnrealSlides>

return in a void functions - pros

- Makes you think about where you leave the function.
- It's consistent with code Visual Studio creates for you.
- You can return earlier than the end (for example an error check fails).



View and comment online at <http://bit.ly/UnrealSlides>

return in a void functions - cons

- It's extra code, and less code is generally better.
- Somebody may write statements below it later, which never get executed.



View and comment online at <http://bit.ly/UnrealSlides>

Write `string GetGuess()`

- Save your code so you can go back.
- Write a function to get the `Guess`
- `return` it as a string.
- Restore your code to its former working glory.



Iterating With `For` & `While` Loops

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- Why we need loops.
- When to use `for` vs `while`.
- The syntax of a `for` loop.
- Think carefully about the first & last loop.
- Write a `for` loop to repeat the game.



View and comment online at <http://bit.ly/UnrealSlides>

When to use **for** vs **while**

- Pick a standard to keep yourself sane, e.g.
- “**Know what you’re in for**” - you know at compile time how many times it will loop.
- “**May be looping for a while**” - you’re not sure how many times it will loop.



View and comment online at <http://bit.ly/UnrealSlides>

The syntax of a **for** loop

```
for (initialization; condition; increase)
    statement;

for (int count = 1; count <= limit; count++)
{
    <the code you want to repeat>
}
```

<http://www.cplusplus.com/doc/tutorial/control>
<https://msdn.microsoft.com/en-us/library/b80153d8.aspx>



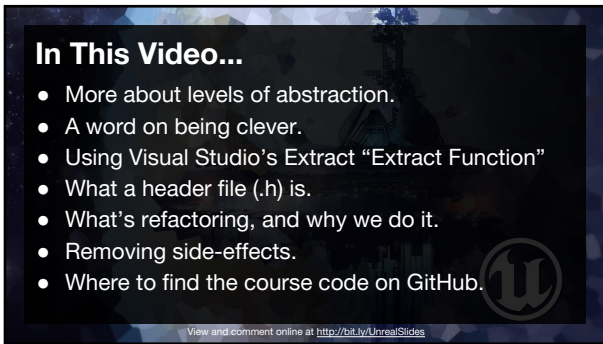
View and comment online at <http://bit.ly/UnrealSlides>

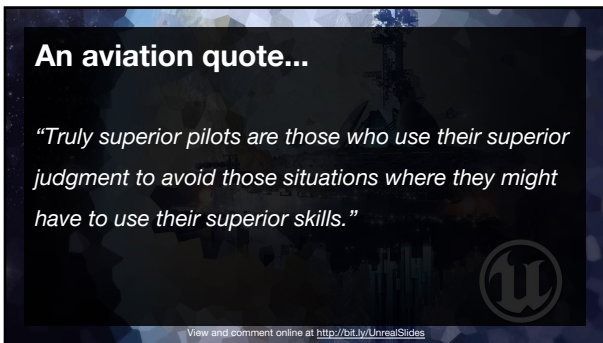
Make the game take 5 guesses

- Use what you’ve learnt so far to make the game take 5 guesses in a row.
- GetGuess() should appear as a function call only once inside a **for** loop.
- Bonus: remember what I said about magic numbers.









Remove the side-effect

- Rename the `GetGuessAndPrintBack()`.
- Move the offending code.
- Test it all still works.
- Are you very happy with how your code reads?



booleans and comparisons

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- What a `bool` is, and how to use it.
- Only use when completely clear what you mean.
- Use `==` for comparison.
- Use `&&` for logical AND.
- Use `||` for logical OR.
- Use `[n]` to access a string, starting at `n=0`.
- Use `' '` for characters, and `" "` for strings.

View and comment online at <http://bit.ly/UnrealSlides>



Write rest of **AskToPlayAgain()**

- Allow for 'y' or 'Y' as the first letter.
- You can ignore the rest of the letters.
- Return true for for yes, false for no*

* This is on the limit of what's "obvious".



Using **do** and **while** in C++

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- What a **do while** loop is.
- How it executes code one or more times.
- Making our game play multiple times.



View and comment online at <http://bit.ly/UnrealSlides>

The syntax of a **do while** loop

```
do {  
    <the code you want to repeat>  
}  
while (condition);
```

The code gets executed once before the check.

<http://www.cplusplus.com/doc/tutorial/control>
<https://msdn.microsoft.com/en-us/library/b0kk5few.aspx>

View and comment online at <http://bit.ly/UnrealSlides>



Make the game play multiple times

- Put a **do while** loop in main.
- Refer to example on previous slide for syntax.
- Test you can play as many times as you like.



Introducing Classes

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- Lookup the Turing machine.
- A quick overview of the MVC pattern.
- User defined types (classes).
- About working at an interface level (black box).
- An overview of `class FBullCowGame`

View and comment online at <http://bit.ly/UnrealSlides>



Read around the topic

Read around these topics...

- Model View Controller (MVC) pattern.
- Turing machines (e.g. Computerphile on YouTube).



Using Header Files as Contracts

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- Introducing .h header files in C++.
- Why the added complexity is worth it.
- Defining the interface to our class.
- Writing our first draft of **FBullCowGame.h**

View and comment online at <http://bit.ly/UnrealSlides>



Write all the methods you can

- Write as many simple signatures as you can
- Don't worry about getting it "right"
- There is no right anyway, the point is to think
- Enjoy working at a higher level.



Including Our Own Header File

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- NEVER use using namespace in a .h
- In fact, why use it at all?
- Create your .cpp files and **#include**
- Don't create chains of includes.



View and comment online at <http://bit.ly/UnrealSlides>

Finish writing blank definitions

- Write blank definitions for all methods.
- Ensure there are no warnings in the .h file.



Instantiating Your Class

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- Relax, they're just user defined types!
- `string FirstName;` creates a string object
- `FBullCowGame BCGame;` works the same way
- These instances are initialised by "constructors"
- Instantiating means "creating an instance of"
- So we're simply creating a game instance.



View and comment online at <http://bit.ly/UnrealSlides>

Create a **BCGame** instance

- Make it the first line of `PlayGame()` for now
- Declare a new object called `BCGame`
- Make its type `FBullCowGame`
- Don't worry about "initialising" it yet
- Make sure you code still runs.



Writing & Using Getter Methods

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- What is a getter method
- Why we never access variables directly
- How to call a method using the dot operator
- Pros and cons of initialising in at compile time
- Using “Rebuild Project” to make VS behave!



View and comment online at <http://bit.ly/UnrealSlides>

Implement **GetCurrentTry()**

- Initialise the value to 1 in the header file (for now)
- Check it works by printing the try from **GetGuess()**
- For example: “**Try 1.** Enter your guess: “
- where 1 is the value of **MyCurrentTry**.



Introducing the **const** Keyword

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- **const**'s meaning depends on context
- Generally means "I promise not to change this"
- What this is depends on exactly where it appears
- At the end of a member function, for example **int GetCurrentTry() const**; it prevents the function from modifying any member variables
- This is a good safety feature.

View and comment online at <http://bit.ly/UnrealSlides>



Apply **const** to all your getters

- Make all getter methods const
- Check it still runs
- Are any of the other functions we've written so far candidates for the use of **const**? If so please suggest which (if any) in the discussions.



Constructors For Initialisation

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- Default constructor called when object created
- Initialize in constructor when decided at runtime
- Initialize in declaration if known at compile time
- Constructor syntax simply: `ClassName();`
- Set the member variables in constructor
- Test this has worked.



View and comment online at <http://bit.ly/UnrealSlides>

Write and test `Reset()`

- Initialise all the member variables
- Set the max tries to 8
- Use the appropriate constant for the magic #
- Check that it works by using `cout` as needed.



Pseudocode Programming

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- More on Pseudocode Programming Practice (PPP)
- Reviewing our code and architecture
- Using `// TODO` as a comment prefix
- Introducing Visual Studio's Task List
- Planning our next wave of coding.

View and comment online at <http://bit.ly/UnrealSlides>



Write your own notes and TODOs

- Go through your code, make sure it makes sense
- Action any existing TODOs that you can
- Write any new TODOs for tasks that must be done
- Do NOT use code in these comments
- Give yourself the gift of working at a higher level.



Using `using` for Type Aliases

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- We're substituting types to be "Unreal ready"
- The declaration is `using <alias> = <type>;`
- For example `using int32 = int;`
- Why Unreal uses `int32` rather than `int`
- `FText` is for output, `FString` is "mutable"
- Where to use each type of string
- Map `FText` and `FString` to `std::string`

View and comment online at <http://bit.ly/UnrealSlides>



Substitute the integer type

- Convert all integers to use the `int32` alias
- Exclude `int main()` as this is called by the OS
- Explicitly substitute at top of file, not via include
- Test your code still runs and reads well.



Using `struct` for Simple Types

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- **struct** is almost identical to **class**
- It's member variables (data) is public by default
- Ideal for simple value types like **BullCowCount**
- Outline **BullCowCount SubmitGuess(FString)**

View and comment online at <http://bit.ly/UnrealSlides>



Pseudocode the function

- Write out the “algorithm” for the function.
- Make your best effort.
- Avoid the use of any code.
- Indent comments inside any loops.



Using **if** Statements in C++

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- Why we need conditionals (selection)
- Use **if** when it reads better (e.g. few conditions)
- Use **switch** for multiple, simple conditions
- (for loads of statements consider a table lookup)
- The syntax of an **if** statement
- Using **if** to write count bulls and cows.



View and comment online at <http://bit.ly/UnrealSlides>

if syntax

```
if (condition1) {  
    statements;  
} else if (condition2) {  
    statements;  
} else {  
    statements;  
}
```

Note this is slightly more compact than Unreal's coding standards.



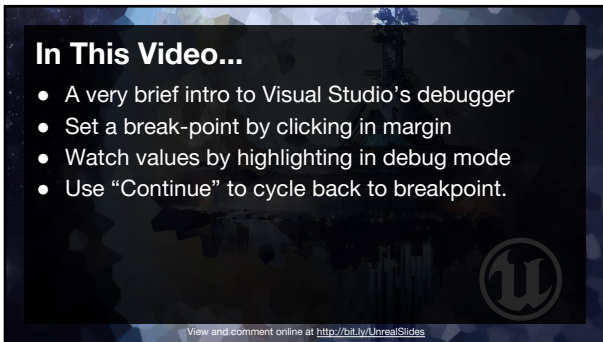
View and comment online at <http://bit.ly/UnrealSlides>

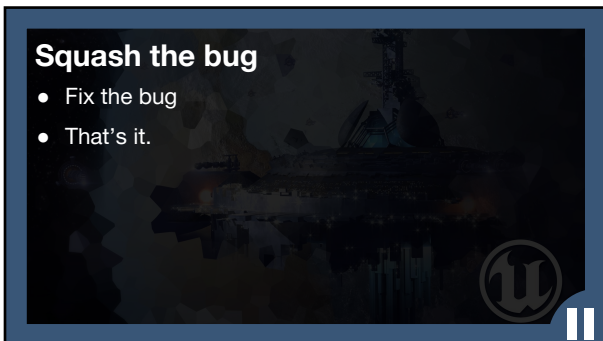
Finish **SubmitGuess()** and test

- Finish the function
- Test it works as expected
- Celebrate your coding ninja skills.

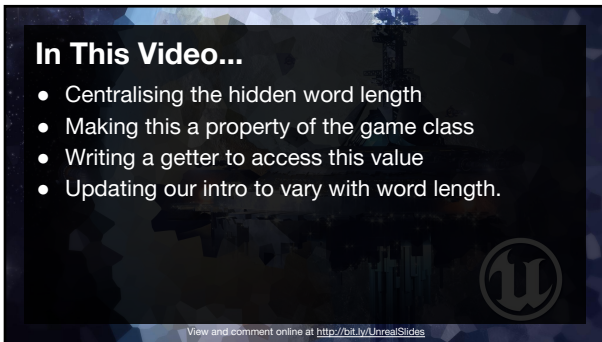


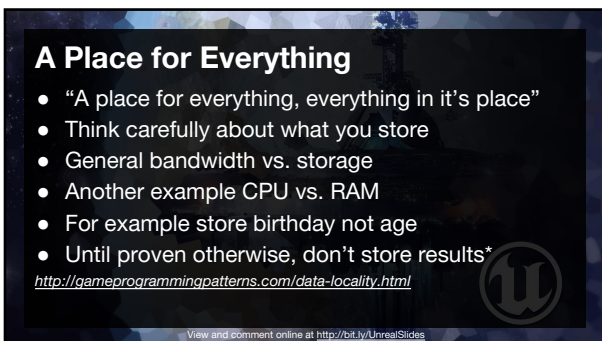












Replace **WORD_LENGTH**

- Write `GetHiddenWordLength()` getter
- Make it read from the current word length
- Do not create another member variable
- Replace **WORD_LENGTH** in `PrintIntro()`
- Test with different hidden word lengths.





In This Video...

- An **enumerated** type consists of named values
- Use instead of coded meaning
- Makes the code more readable and meaningful
- Only defined values can be used - more robust
- A benefit of C++ 11's strongly typed enums
- Creating an **enum class** for error checking.



View and comment online at <http://bit.ly/UnrealSlides>

Finish the error value list

- Write at least two more error values
- Think about what could break the game
- ... or try weird input and see what DOES break



Writing Error Checking Code

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- Use **else if** for the first time
- Outline or **CheckGuessValidity()** method
- Write working code for checking guess length
- Use the debugger to test the return values.



View and comment online at <http://bit.ly/UnrealSlides>

Write the word length check

- No need to call any “helper methods”
- Use `!=` for “not equals”
- Call `GetHiddenWordLength()`
- Test the value using the debugger.



Using `switch` Statements

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- Use our error values to communicate with user
- All our user interaction is via `GameManager.cpp`
- We'll use `FText` in this file, as it's UI text
- We can “switch” what we say based on the error
- The syntax of a `switch` statement
- Remember your `break` keywords!



View and comment online at <http://bit.ly/UnrealSlides>

switch statement syntax

```
switch (expression) // expression is what we switch based on
{
    case constant1:
        statement(s);
        break;
    case constant2:
        statement(s);
        break;
    default:
        statement(s);
}
```

View and comment online at <http://bit.ly/UnrealSlides>



Write the rest of the errors

- Write a message to the user for each error
- Tell them how to get it right next time
- Remember your **break** statements!
- Test the console output for wrong word length.



Warm Fuzzy Feelings

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- *Don't* get comfortable with compiler warnings
- Refactor `GetValidGuess()` to remove warning
- Rename `SubmitGuess()` to `SubmitValidGuess()`
- Improve readability of `SubmitValidGuess()`
- Get a warm fuzzy feeling!

View and comment online at <http://bit.ly/UnrealSlides>



Try and remove the warning

- Try and remove the warning
- Check your code still runs
- Watch my suggested method.



Handling Game Win Condition

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- Change our `PlayGame()` loop to a `while`
- Implement our `IsGameWon()` function

View and comment online at <http://bit.ly/UnrealSlides>



Finish `IsGameWon()`

- Define the appropriate getter method
- Create a private variable, prefixed with `b`
- Set the private variable in `SubmitValidGuess()`
- Test you can now win the game.



Win or Lose "Screen"

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- Write a method to print a game summary to the screen once the game is over.

View and comment online at <http://bit.ly/UnrealSlides>



Define `PrintGameSummary()`

- Arrange for a "You won / bad luck" message
- Decide where in the program it goes
- Write a function for it
- Test it works.



Introducing Big O Notation

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- Algorithm: the recipe for solving a problem
- or: 45th US Vice President's dance style
- Introducing the complexity of algorithms
- A quick introduction to "Big O" notation
- Comparing three ways of checking for isograms.



View and comment online at <http://bit.ly/UnrealSlides>

What order is **IsIsogram()** at best?

- Vote for $O(n)$, $O(n \log n)$ or $O(n^2)$
- Share your vote in the discussions
- Explain why you think that's as fast as possible
- Carry on watching.



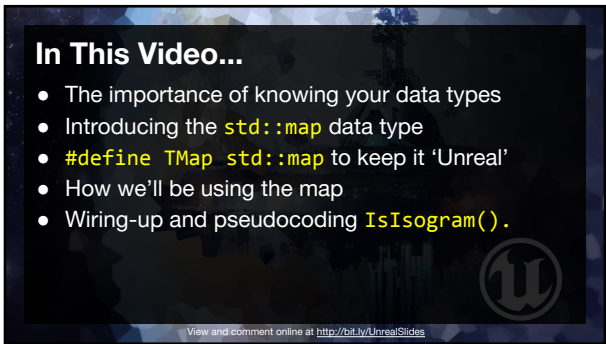
Further reading

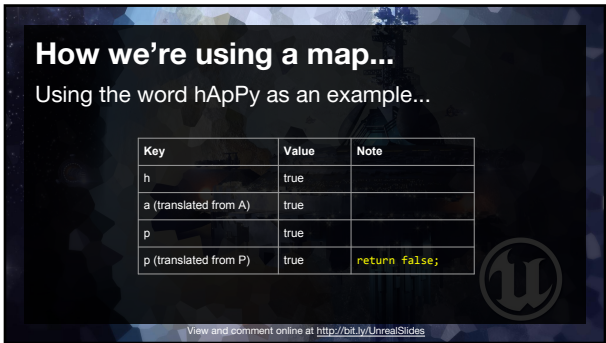
- <http://stackoverflow.com/questions/9107516/sorting-characters-of-a-c-string>
- <http://en.cppreference.com/w/cpp/algorithm/sort>
- <http://www.wolframalpha.com/input/?i=n%5E2+vs+%28n+log+n%29+vs+n%2C+n%3D2+to+17>
- https://en.wikipedia.org/wiki/Sorting_algorithm
- https://en.wikipedia.org/wiki/Big_O_notation
- <https://en.wikipedia.org/wiki/Isogram>



View and comment online at <http://bit.ly/UnrealSlides>







“Wire-up” **IsIsogram()**

- Create a private *function*
- Decide if it should be **const** or not
- Call it from **CheckGuessValidity()***
- Simply return true for now inside **IsIsogram()**

Hint: you may need a not ! operator.



Pseudocode **IsIsogram()**

- Only write comments for now
- Indent the comments as necessary
- Avoid code in the comments, work at a high level
- Run through mentally with some examples.



Range-based **for** Loop

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- Introducing containers and iterators
- Using a range-based **for** loop in Unreal*
- Gently introducing the **auto** keyword
- Finishing our **IsIsogram()**

* <https://www.unrealengine.com/blog/range-based-for-loops>



View and comment online at <http://bit.ly/UnrealSlides>

std::map syntax

- `TMap<char, bool> LetterSeen;` to declare
- Using `LetterSeen[Letter]` to access
- You can assign to the map element
- e.g. `LetterSeen[Letter] = true;`

`std::unordered_set` or Unreal `TSet` is a valid alternative. Map is a more versatile.



View and comment online at <http://bit.ly/UnrealSlides>

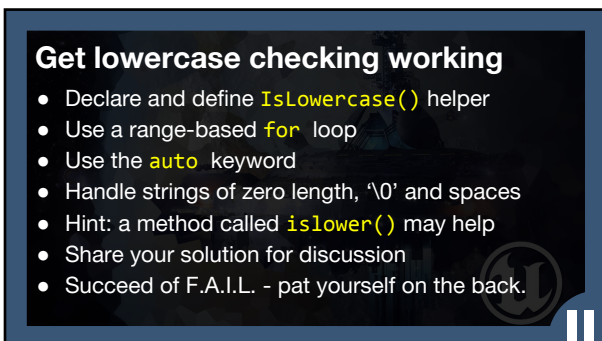
Finish **IsIsogram()**

- Finish the function
- Test it thoroughly
- Try entering just `\0` as a guess
- Try a blank string
- Try mixed case e.g. `Aa`
- Share your implementation in the discussions.









Avoid Implicit dependencies

Implicit: suggested though not directly expressed.

Dependent: reliant on something else.

An implicit dependency is when it's not completely clear that one thing depends on another. For example the order of checks in `CheckGuessValidity()`.

View and comment online at <http://bit.ly/UnrealSlides>



Playtesting Your Game

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- Having someone else play test your game is vital
- Silently take notes, or record screen if possible
- Immediately go away and fix obvious bugs
- For improvements consider 2nd or 3rd opinion
- Repeat until the bug / issue rate plateaus.

View and comment online at <http://bit.ly/UnrealSlides>



Playtest the game

- Find somebody else, Skype if needed
- Share the bugs / improvements in Discussions
- Make it clear which version you're commenting on
- Also play-test my game, download from...

https://github.com/UnrealCourse/02_BullCowGame/archive/master.zip



My playtesting actions (thanks Lizzie)

- Remove new line before bull cow count
- Tell user how many guesses they have left
- All a bit "left brain", needs some ASCII art
- Spelling error in "Better luck next time!"

View and comment online at <http://bit.ly/UnrealSlides>



Difficulty & Play Tuning

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- About the flow channel*
- **map** word length to max tries
- Play test to determine correct difficulty.

* Read more in Sylvester, T. *Designing Games* - O'Reilly

View and comment online at <http://bit.ly/UnrealSlides>



Map word length to max tries

- Populate your word length : max tries map
- Do more play testing to find the "sweet spot"
- Do this for at least 4-6 letter words
- Optionally: let user select word length.



Polishing & Packaging

@UnrealCourse :: www.UnrealCourse.com



In This Video...

- First impressions count (think reviews)
- Don't ship a half-baked product, even if digital
- Check through your code (polish)
- Ship to your customers (package).



View and comment online at <http://bit.ly/UnrealSlides>

Polish your game

- Comment with “why”, remove obvious comments
- Introduce your classes with block comments
- Use **#pragma once** at the top of each file
- Deal with or delete any TODO items
- Capture future ideas / improvements.

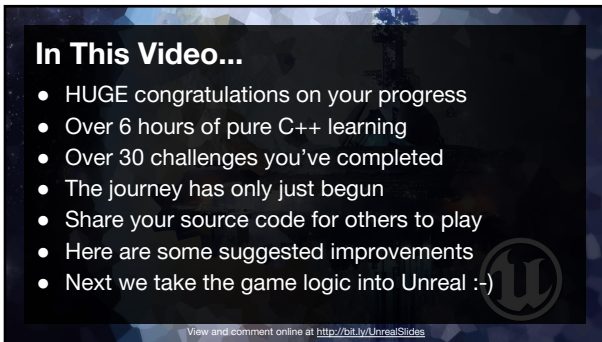


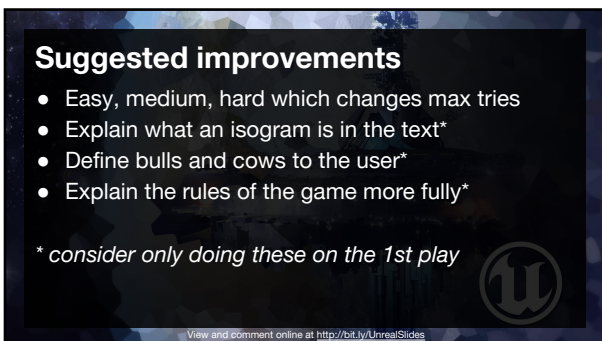
Ship your game

- Zip up **source code** (not **.exe** or **.app**) and share in discussions via DropBox, Drive, YouSendIt etc
- Ensure code is in runnable state and archive
- Celebrate your success - seriously.

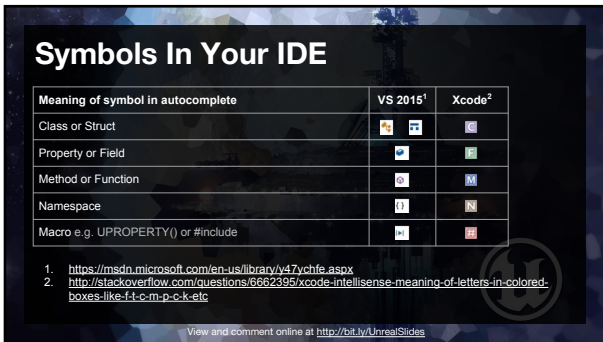


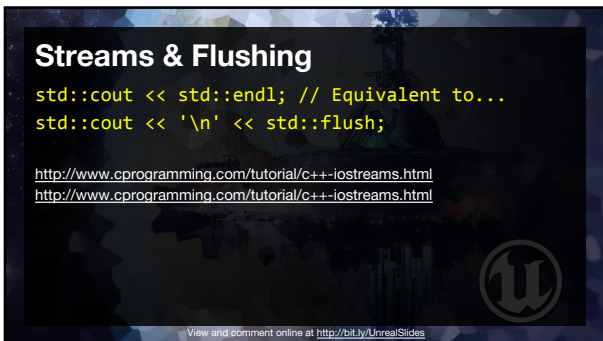












Parameter vs. Argument

- They are two sides of the same coin
- They often get used interchangeably, but strictly...
- You **define parameters**
- You call with **argument(s)**

<http://stackoverflow.com/questions/156767/whats-the-difference-between-an-argument-and-a-parameter>

View and comment online at <http://bit.ly/UnrealSlides>



Unreal Coding Standards

- We will use a slightly different { } strategy.
- This is to save vertical space on the screen.
- Otherwise we'll generally stick to these*

*<https://docs.unrealengine.com/latest/INT/Programming/Development/CodingStandard/index.html>

View and comment online at <http://bit.ly/UnrealSlides>



Unreal's TSet & std::unordered_set

"Unordered sets are containers that store unique elements in no particular order, and which allow for fast retrieval of individual elements based on their value."

This is ideal alternative for our **SeenLetters** container. Thanks to Daniel for bringing this up.

View and comment online at <http://bit.ly/UnrealSlides>



Braced Initializers

- `int32 AgeInYears {24};` initialisation syntax
- Prevents “narrowing conversions” e.g.
- `int32 AgeInYears {24.3};` ⇐ error
- `int32 AgeInYears = 24.3;` ⇐ narrows to 24
- Defaults to the right empty value
- e.g. `char PreviousChar {}`
- Compatible with lists and other containers.

View and comment online at <http://bit.ly/UnrealSlides>