CMP105 Games Programming

Enumeration

This week

- Enumeration
- Game states
  - Use of enumerations

Enumeration

- Enumerations are a grouping of constants
- Commonly referred to as an enum
- Like classes each enum defines a new type
- Two major types of enum
  - Scoped
    - Follows the normal rules of scoping, elements are inaccessible outside the scope of the enumerations
    - Defined by enum class
  - Unscoped
    - Elements placed in the same scope as the enumeration itself
    - Defined by enum

Scoped

```
enum class Suit {Diamonds, Hearts, Clubs, Spades};

void playCard(Suit suit)
{
        if(suit == Suit::Diamonds)
        {
                //...
        }
}
```

Unscoped

```
enum Suit {Diamonds, Hearts, Clubs, Spades};

void playCard(Suit suit)
{
        if(suit == Clubs)
        {
                // ...
        }
}
```

Enumeration values

- Enums auto assign a value to the elements starting at zero
- Incrementing by 1 greater than the preceding value

```
enum Suit {Diamonds, Hearts, Clubs, Spades};
                0         1       3      4
```

Enumeration values

- However, you can specify values for one or more of the elements
- Elements do not need to have unique values

```
enum Suit {Diamonds = 7, Hearts = 5, Clubs = 2, Spades = 1};
```

- Or

```
enum Suit {Diamonds = 1, Hearts, Clubs, Spades};
```

Enum types

- Unscoped enums are implicitly converted to integral types
- Scoped enums are not

```
int i = Diamonds;            // OK
int j = Suit::Diamonds;      // Bad
```

Enum types

- Enums can have a type specified
- New to C++11

```
enum direction : char { left = 'l', right = 'r' };
enum Suit : int {Diamonds, Hearts, Clubs, Spades};
```

Game states

- A game state is one of the many different layers of your game
  - Intro
  - Main menu
  - Game / level
  - Credits
  - etc

Game states

- A simple method of tracking game states is using an enum of possible game states
- Paired with a switch statement to control what happens based on the current game state

```
enum class GameState {MENU, LEVEL, CREDITS};
```

```
switch (state)
{
case (GameState::MENU) :
        menu.handleInput(deltaTime);
        menu.update(deltaTime);
        menu.render();
        state = menu.getState();
        break;

case(GameState::LEVEL):
        game.handleInput(deltaTime);
        game.update(deltaTime);
        game.render();
        state = game.getState();
        break;

case(GameState::CREDITS) :
        //...
        break;
}
```

Live demo

- Enums and game states in action

Summary

- Warning
  - This will work well with the small games but on larger games becomes unmanageable

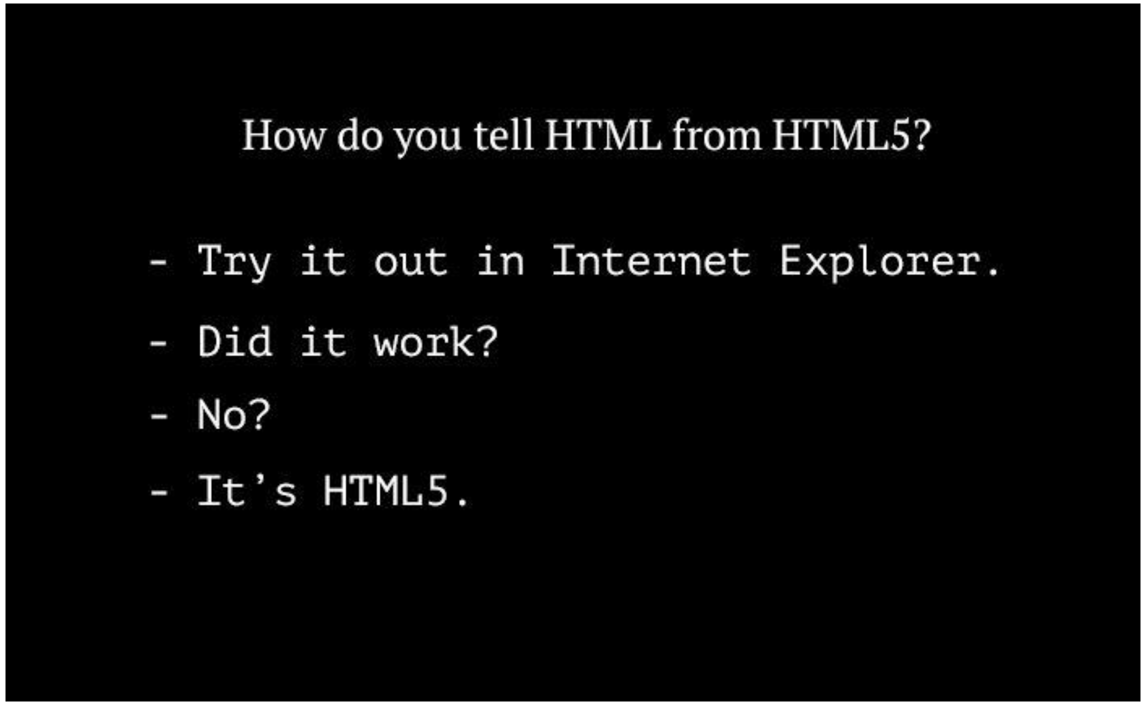
- Other uses of enums
  - Types of Sprite (world, bullet, player, enemy etc)
    - Useful during collision detection
  - Character or sprite state
    - State; JUMPING, DUCKING, DEAD
    - Instead of maintaining a large number booleans

Important information

- Next two weeks are a holiday
  - 27th March – 7th April 2017
- Normal classes resume Monday 10th April 2017

In the labs

- Working with Enums and game states
- Working on coursework



How do you tell HTML from HTML5?

```
- Try it out in Internet Explorer.
- Did it work?
- No?
- It's HTML5.
```

- Further reading
  - http://www.learncpp.com/cpp-tutorial/4-5a-enum-classes/