Game Tools and the command line

- many games have in game tools
  - you are working on one!
  - many level editors exist etc
- sometimes it is useful to develop game tools outside the game
  - provides mechanisms to batch process/script cpu intensive activities which can be run on server class machines
- there is benefit from understanding how such tools operate as this knowledge will be usefully transferable

Command line tools

- often have similar options
  - most GNU utilities will implement
  - `-h`, `--version` options (for help and version)
- nearly all command line utilities also have on system man pages associated with them

Examine the compilers on GNU/Linux

- the `gcc` compiler is an interesting example, and while `gcc` is a tool (C compiler).
  - the GCC project is the GNU Compiler Collection
  - which consists of the C compiler, C++ compiler, Java compiler, (and Modula-2 compiler)
- here we will look at some of the common options to `gcc`, `g++` and `gm2`
- notice how these three programs exist and GCC have tried to unify these command options
- these slides are simply a taster and huge simplification of how GCC might be used

GCC debugging

- all front ends (in our case: `gcc`, `g++` and `gm2`) accept `-g -O0` which tell the compiler not to optimise and emit debugging information for `gdb`

GCC uniformity of options

- they all understand `--version`
- ```
  $ gcc --version
  gcc (Debian 6.3.0-18+deb9u1) 6.3.0 20170516
  Copyright (C) 2016 Free Software Foundation, Inc.
  This is free software; see the source for copying conditi
  warranty; not even for MERCHANTABILITY or FITNESS FOR A P
  ```


GCC uniformity of options

- ```
  $ g++ --version
  g++ (Debian 6.3.0-18+deb9u1) 6.3.0 20170516
  Copyright (C) 2016 Free Software Foundation, Inc.
  This is free software; see the source for copying conditi
  warranty; not even for MERCHANTABILITY or FITNESS FOR A P
  ```
- ```
  $ gm2 --version
  gm2 (GCC) 6.4.0
  Copyright (C) 2017 Free Software Foundation, Inc.
  This is free software; see the source for copying conditi
  warranty; not even for MERCHANTABILITY or FITNESS FOR A P
  ```


GCC uniformity of options

- simple compiles can be performed by:
- ```
  $ gcc -o outputbinary -g sourcefile.c
  ```
- the `-g` option means generate debugging information so your program can be debugged via `gdb`

GCC uniformity of options

- for complex projects can be compiled and linked via:
- ```
  $ gcc -g -c sourcefile1.c
  $ gcc -g -c sourcefile2.c
  $ gcc -g -c sourcefile3.c
  $ gcc -o outputbinary sourcefile1.o sourcefile2.o sourcef
  ```
  ```

GCC uniformity of options

- there are many options in these tools, only a few of which are mentioned in these slides
- turn on all warnings by: `-Wall`
- so our command line to compile `hello.c` is:
- ```
  $ gcc -g -O0 -Wall -c hello.c
  ```


GCC uniformity of options

- notice that this compiles `hello.c` but does not link it
- to link this we can:
- ```
  $ gcc -g hello.o
  ```
- the default output file is: `a.out` (which is used if you omit the `-o` option)

GCC uniformity of options

- we could combine the last two steps by:
- ```
  $ gcc -g -O0 -Wall hello.c
  ```
- many tools try and provide consistency between options
  - learn once use many times
  - many options have become a de facto standard `--version`
- it is important to adhere to de facto standards when producing useful tools

Coursework file format structure

- examine the file structure of the coursework zip file
- ```
  $ ls
  Level1   Level2   Level3
  $ ls Level1
  Enemy.bmp
  fire.bmp
  Floor.bmp
  Goal.bmp
  Level.txt
  Player.bmp
  Wall.bmp
  ```

Level1.txt

```
Height=12                       0..Y axis limit-1
Width=12                        0..X axis limit-1
FFFFWFFFWFFF                    row 0
FFFFWFFFFFFF
FFFFWWWFWWWF
FFFFWFFFFFWF
WWWFWFWFWFWF
FFFFWFWFWFWF
FFFFWFWFWFWF
FFFFWFWFWFWF
FFFFWFWFFFWF
FWWWWFFFWWWF
FFFFWFWWWFFF
FFFFFFWFFFFF                    row 11
StartPosition={0, 0}            start coordinate
GoalPosition={11, 11}           goal coordinate
TimeToComplete=20               time to finish the level
NumEnemies=9                    number of enemies
```


Level1.txt

```
{7, 3}                          coord of enemy number 0
{5, 8}                          coord of enemy number 1
{3, 7}
{0, 6}
{3, 10}
{9, 5}
{9, 10}
{11, 0}
{3, 3}                          coord of enemy number 8
NumFire=9                       number of fireballs
{8, 1}                          coord of fireball 0
{3, 4}
{0, 9}
{4, 11}
{6, 9}
{6, 3}
{11, 9}
{11, 3}
{5, 4}                          coord of fireball 8
```


Level1.txt

- notice how Ben is using a text file to describe the level
  - also interesting to examine the data structures in the previous weeks slides
- consider the declarations found in `JewelThief/CLevel.cs`

Level1.txt

JewelThief/CLevel.cs

```
namespace WizardDungeon
{
    /// <summary>
    /// This enum characterises the type a tile can be
    /// in the game (e.g. wall or floor).
    /// </summary>
    enum eTileType
    {
        Wall,
        Floor
    };
```

Level1.txt

- **JewelThief/CLevel.cs**

```
    /// <summary>
    /// This enum defines whether a level should be flipp
    /// vertically or horizontally.
    /// </summary>
    public enum eFlipDirection
    {
        Horizontal,
        Vertical
    };
```


Level1.txt

- **JewelThief/CLevel.cs**

```
  /// This class is used to store all of the information
  /// a level.
  /// This includes the level layout stored in m_levelMap
  /// start positions of the player and the enemies and t
  /// goal.
  class CLevel
  {
    public int Height;
    public int Width;
    public int Time;
    public CPoint2i StartPosition { get; set; } = new CPo
    public CPoint2i GoalPosition  { get; set; } = new CPo
    public List<CPoint2i>  EnemyPositions { get; set; } =
    public List<CPoint2i>  FirePositions { get; set; } =

    private eTileType[,] m_levelMap = null;
```


Level1.txt

- examine file `JewelThief/CLevel.cs` for the constructors and further details
- we can easily see that the txt file is loaded into these data structures
- text files are often used to configure applications (this is the norm for GNU/Linux and Unix utilities)