HELLO WORLD: YOUR FIRST PROGRAM
Topics
Topics

- Hello World?
Topics

- Hello World?
- Creating a Unity Project
Topics

- Hello World?
- Creating a Unity Project
  – The Unity Project Folder
Topics

- Hello World?
- Creating a Unity Project
  - The Unity Project Folder
- MonoDevelop: Unity's Code Editor
Topics

- Hello World?
- Creating a Unity Project
  - The Unity Project Folder
- MonoDevelop: Unity's Code Editor
- Attaching Scripts to GameObjects
Topics

- Hello World?
- Creating a Unity Project
  - The Unity Project Folder
- MonoDevelop: Unity's Code Editor
- Attaching Scripts to GameObjects
- Start() and Update()
Topics

- Hello World?
- Creating a Unity Project
  - The Unity Project Folder
- MonoDevelop: Unity's Code Editor
- Attaching Scripts to GameObjects
- Start() and Update()
- GameObject Prefabs and Instantiation
Topics

- Hello World?
- Creating a Unity Project
  - The Unity Project Folder
- MonoDevelop: Unity's Code Editor
- Attaching Scripts to GameObjects
- Start() and Update()
- GameObject Prefabs and Instantiation
- The HelloWorld Project
Hello World?
Hello World?

- Hello World is often the first program written by anyone learning a new programming language.
Hello World?

- Hello World is often the first program written by anyone learning a new programming language.
- Outputs "Hello World!" to the Console
Hello World?

- Hello World is often the first program written by anyone learning a new programming language.
- Outputs "Hello World!" to the Console
Hello World?
Hello World?

- The code of HelloWorld.cs is very simple:
Hello World?

- The code of HelloWorld.cs is very simple:
Creating a Unity Project
Creating a Unity Project

- From the menu bar, choose *File > New Project*...
Creating a Unity Project

- From the menu bar, choose *File > New Project…*
- Choose the location for your project folder
Creating a Unity Project

- From the menu bar, choose *File > New Project...*
- Choose the location for your project folder
  - Mac OS X
Creating a Unity Project

- From the menu bar, choose *File > New Project*…
- Choose the location for your project folder
  - Mac OS X
    - Click the *Set*… button
Creating a Unity Project

- From the menu bar, choose *File > New Project*…
- Choose the location for your project folder
  - *Mac OS X*
    - Click the *Set*… button
    - Navigate to the right location
Creating a Unity Project

- From the menu bar, choose File > New Project…
- Choose the location for your project folder
  - Mac OS X
    - Click the Set… button
    - Navigate to the right location
    - Type the project name into the Save As field
Creating a Unity Project

- From the menu bar, choose File > New Project…
- Choose the location for your project folder
  - Mac OS X
    • Click the Set… button
    • Navigate to the right location
    • Type the project name into the Save As field
    • Click the Save button
Creating a Unity Project

- From the menu bar, choose File > New Project…
- Choose the location for your project folder
  - Mac OS X
    - Click the Set… button
    - Navigate to the right location
    - Type the project name into the Save As field
    - Click the Save button
  - Windows
Creating a Unity Project

- From the menu bar, choose *File > New Project*…
- Choose the location for your project folder
  - **Mac OS X**
    - Click the *Set*… button
    - Navigate to the right location
    - Type the project name into the *Save As* field
    - Click the *Save* button
  - **Windows**
    - Click the *Browse*… button
Creating a Unity Project

- From the menu bar, choose File > New Project…
- Choose the location for your project folder
  - Mac OS X
    - Click the Set… button
    - Navigate to the right location
    - Type the project name into the Save As field
    - Click the Save button
  - Windows
    - Click the Browse… button
    - Navigate to the right location
Creating a Unity Project

- From the menu bar, choose *File > New Project…*
- Choose the location for your project folder
  - **Mac OS X**
    - Click the *Set…* button
    - Navigate to the right location
    - Type the project name into the *Save As* field
    - Click the *Save* button
  - **Windows**
    - Click the *Browse…* button
    - Navigate to the right location
    - Click the *New Folder* button and give the new folder the name of your project.
Creating a Unity Project

- From the menu bar, choose *File > New Project*…
- Choose the location for your project folder
  - Mac OS X
    - Click the *Set*… button
    - Navigate to the right location
    - Type the project name into the *Save As* field
    - Click the *Save* button
  - Windows
    - Click the *Browse*… button
    - Navigate to the right location
    - Click the *New Folder* button and give the new folder the name of your project.
    - Click the *Select Folder* button
Creating a Unity Project
Creating a Unity Project

- Set up defaults for 3D
Creating a Unity Project

- Set up defaults for 3D
- Click the *Create Project* or *Create* button
Creating a Unity Project

- Set up defaults for 3D
- Click the *Create Project* or *Create* button
- Appendix A contains detailed instructions
Creating a Unity Project
Creating a Unity Project

- The Project pane shows the contents of the Assets folder inside your Unity project folder
Creating a Unity Project

- The Project pane shows the contents of the Assets folder inside your Unity project folder
  - Right-click in the Project pane and choose Reveal in Finder (or Show in Explorer) from the pop-up menu
Creating a Unity Project

- The Project pane shows the contents of the Assets folder inside your Unity project folder
  - Right-click in the Project pane and choose *Reveal in Finder* (or *Show in Explorer*) from the pop-up menu
Creating a Unity Project

- The Project pane shows the contents of the Assets folder inside your Unity project folder
  - Right-click in the Project pane and choose Reveal in Finder (or Show in Explorer) from the pop-up menu
MonoDevelop: Unity's Code Editor
MonoDevelop: Unity's Code Editor

- Unity uses MonoDevelop for code editing
MonoDevelop: Unity's Code Editor

- Unity uses MonoDevelop for code editing
  - MonoDevelop is a separate program developed by a different team
MonoDevelop: Unity's Code Editor

- Unity uses MonoDevelop for code editing
  - MonoDevelop is a separate program developed by a different team

- To open MonoDevelop, double-click any C# script in your Project pane
MonoDevelop: Unity's Code Editor

- Unity uses MonoDevelop for code editing
  - MonoDevelop is a separate program developed by a different team
- To open MonoDevelop, double-click any C# script in your Project pane
  - This will launch MonoDevelop
MonoDevelop: Unity's Code Editor

- **Unity uses MonoDevelop for code editing**
  - MonoDevelop is a separate program developed by a different team

- **To open MonoDevelop, double-click any C# script in your Project pane**
  - This will launch MonoDevelop
  - Though the launch process takes some time
MonoDevelop: Unity's Code Editor

- Unity uses MonoDevelop for code editing
  - MonoDevelop is a separate program developed by a different team

- To open MonoDevelop, double-click any C# script in your Project pane
  - This will launch MonoDevelop
  - Though the launch process takes some time

- You must save a document in MonoDevelop for it to recompile and update in Unity
MonoDevelop: Unity's Code Editor

- Unity uses MonoDevelop for code editing
  - MonoDevelop is a separate program developed by a different team

- To open MonoDevelop, double-click any C# script in your Project pane
  - This will launch MonoDevelop
  - Though the launch process takes some time

- You must save a document in MonoDevelop for it to recompile and update in Unity

- On Windows, Microsoft Visual Studio may be used
MonoDevelop: Unity's Code Editor

- Unity uses MonoDevelop for code editing
  - MonoDevelop is a separate program developed by a different team
- To open MonoDevelop, double-click any C# script in your Project pane
  - This will launch MonoDevelop
  - Though the launch process takes some time
- You must save a document in MonoDevelop for it to recompile and update in Unity
- On Windows, Microsoft Visual Studio may be used
  - Instructions for this can be found online
MonoDevelop: Unity's Code Editor

The MonoDevelop Window
MonoDevelop: Unity's Code Editor

The MonoDevelop Window

Code Pane
MonoDevelop: Unity's Code Editor

The MonoDevelop Window

Classes Pane

Code Pane
MonoDevelop: Unity's Code Editor

The MonoDevelop Window

Classes Pane

Document Outline Pane

Code Pane

The MonoDevelop Window
Attaching Scripts to GameObjects

To work in Unity, a C# script must be attached to a GameObject
Attaching Scripts to GameObjects

This makes the script a *component* of the GameObject.
Start() and Update()
Start() and Update()

- You make use of Start() and Update() in the HelloWorld project
Start() and Update()

- You make use of Start() and Update() in the HelloWorld project
  - void Start() {...}
Start() and Update()

- You make use of Start() and Update() in the HelloWorld project
  - void Start() {...}
    - Called once
Start() and Update()

- You make use of Start() and Update() in the HelloWorld project
  - void Start() {...}
    - Called once
    - Called immediately before the first Update() is called
Start() and Update()

- You make use of Start() and Update() in the HelloWorld project
  - `void Start() {...}
    • Called once
    • Called immediately before the first Update() is called
  - `void Update() {...}
You make use of Start() and Update() in the HelloWorld project

- void Start() {…}
  - Called once
  - Called immediately before the first Update() is called

- void Update() {…}
  - Called every frame
You make use of Start() and Update() in the HelloWorld project

- void Start() {...}
  - Called once
  - Called immediately before the first Update() is called

- void Update() {...}
  - Called every frame
  - This can happen over 200 times per second!
Start() and Update()

- You make use of Start() and Update() in the HelloWorld project
  - `void Start() {...}`
    - Called once
    - Called immediately before the first Update() is called
  - `void Update() {...}`
    - Called every frame
    - This can happen over 200 times per second!
  - `void Awake() {...}` (not used in HelloWorld, but important)
Start() and Update()

- You make use of Start() and Update() in the HelloWorld project
  - void Start() {...}
    - Called once
    - Called immediately before the first Update() is called
  - void Update() {...}
    - Called every frame
    - This can happen over 200 times per second!
  - void Awake() {...} (not used in HelloWorld, but important)
    - Called once
You make use of `Start()` and `Update()` in the HelloWorld project

- **void Start() {…}**
  - Called once
  - Called immediately before the first `Update()` is called

- **void Update() {…}**
  - Called every frame
  - This can happen over 200 times per second!

- **void Awake() {…}** (not used in HelloWorld, but important)
  - Called once
  - Called at the moment the GameObject is created
You make use of Start() and Update() in the HelloWorld project

• void Start() {…}
  • Called once
  • Called immediately before the first Update() is called

• void Update() {…}
  • Called every frame
  • This can happen over 200 times per second!

• void Awake() {…} (not used in HelloWorld, but important)
  • Called once
  • Called at the moment the GameObject is created
  • Guaranteed to be called before Start()
GameObject Prefabs and Instantiation
 GameObject Prefabs and Instantiation

- A *prefab* is a mold from which GameObject *instances* can be made
GameObject Prefabs and Instantiation

- A *prefab* is a mold from which GameObject *instances* can be made
  - Created by dragging a GameObject from the Hierarchy pane into the Project pane
GameObject Prefabs and Instantiation

- A *prefab* is a mold from which GameObject *instances* can be made
  - Created by dragging a GameObject from the Hierarchy pane into the Project pane
  - Can be assigned to a script variable in the Inspector pane
A prefab is a mold from which GameObject instances can be made

- Created by dragging a GameObject from the Hierarchy pane into the Project pane
- Can be assigned to a script variable in the Inspector pane

```csharp
public GameObject gameObjectPrefab;
```
GameObject Prefabs and Instantiation

- A *prefab* is a mold from which GameObject *instances* can be made
  - Created by dragging a GameObject from the Hierarchy pane into the Project pane
  - Can be assigned to a script variable in the Inspector pane
    - `public GameObject gameObjectPrefab;`
  - Then, an instance of the prefab can be created in code
GameObject Prefabs and Instantiation

- A *prefab* is a mold from which GameObject *instances* can be made
  - Created by dragging a GameObject from the Hierarchy pane into the Project pane
  - Can be assigned to a script variable in the Inspector pane
    - `public GameObject gameObjectPrefab;`
  - Then, an instance of the prefab can be created in code
    - `Instantiate( gameObjectPrefab );`
GameObject Prefabs and Instantiation

- **A prefab** is a mold from which GameObject *instances* can be made
  - Created by dragging a GameObject from the Hierarchy pane into the Project pane
  - Can be assigned to a script variable in the Inspector pane
    - ```csharp
        public GameObject gameObjectPrefab;
    ```
  - Then, an instance of the prefab can be created in code
    - ```csharp
        Instantiate( gameObjectPrefab );
    ```

- This is used in HelloWorld to create thousands of instances of a Cube GameObject prefab
The HelloWorld Project
The HelloWorld Project

- Output "Hello World!" to the Console pane
The HelloWorld Project

- Output "Hello World!" to the Console pane
  - Once using Start()
The HelloWorld Project

- Output "Hello World!" to the Console pane
  - Once using Start()
  - Many times using Update()
The HelloWorld Project

- Output "Hello World!" to the Console pane
  - Once using Start()
  - Many times using Update()

- Create a Cube prefab that reacts to gravity & physics
The HelloWorld Project

- Output "Hello World!" to the Console pane
  - Once using Start()
  - Many times using Update()

- Create a Cube prefab that reacts to gravity & physics

- Instantiate an instance of the Cube prefab
The HelloWorld Project

- Output "Hello World!" to the Console pane
  - Once using Start()
  - Many times using Update()
- Create a Cube prefab that reacts to gravity & physics
- Instantiate an instance of the Cube prefab
  - Once using Start()
The HelloWorld Project

- Output "Hello World!" to the Console pane
  - Once using Start()
  - Many times using Update()

- Create a Cube prefab that reacts to gravity & physics

- Instantiate an instance of the Cube prefab
  - Once using Start()
  - Many times using Update()
The HelloWorld Project

- **Output "Hello World!" to the Console pane**
  - Once using `Start()`
  - Many times using `Update()`

- **Create a Cube prefab that reacts to gravity & physics**

- **Instantiate an instance of the Cube prefab**
  - Once using `Start()`
  - Many times using `Update()`
    - This will create a cascade of thousands of Cube instances
The HelloWorld Project

- Output "Hello World!" to the Console pane
  - Once using Start()
  - Many times using Update()

- Create a Cube prefab that reacts to gravity & physics

- Instantiate an instance of the Cube prefab
  - Once using Start()
  - Many times using Update()
    - This will create a cascade of thousands of Cube instances
  - Over other physically-modeled objects
The HelloWorld Project

The final HelloWorld scene
Chapter 18 – Summary
Chapter 18 – Summary

- Hello World is a common first program to make in any new language
Chapter 18 – Summary

- Hello World is a common first program to make in any new language
- Unity projects are stored as many files in project folders on your hard drive
Chapter 18 – Summary

- Hello World is a common first program to make in any new language
- Unity projects are stored as many files in project folders on your hard drive
- MonoDevelop is used to edit code for Unity
Chapter 18 – Summary

- Hello World is a common first program to make in any new language
- Unity projects are stored as many files in project folders on your hard drive
- MonoDevelop is used to edit code for Unity
- Scripts must be attached to GameObjects to run
Chapter 18 – Summary

- Hello World is a common first program to make in any new language
- Unity projects are stored as many files in project folders on your hard drive
- MonoDevelop is used to edit code for Unity
- Scripts must be attached to GameObjects to run
- Start(), Update(), and Awake() are called at different times and have different uses
Chapter 18 – Summary

- Hello World is a common first program to make in any new language
- Unity projects are stored as many files in project folders on your hard drive
- MonoDevelop is used to edit code for Unity
- Scripts must be attached to GameObjects to run
- Start(), Update(), and Awake() are called at different times and have different uses
- GameObject prefabs can be instantiated many times