Computer Game Programming

แนวคิดการเขียนโปรแกรม

การเขียนโปรแกรมคอมพิวเตอร์เป็นพื้นฐานในการเขียนโปรแกรมและการพัฒนาเกมคอมพิวเตอร์ การเข้าใจหลักการเขียนโปรแกรมคอมพิวเตอร์ที่มีความจำเป็นอย่างมาก เพื่อให้ผลลัพธ์จากการพัฒนาโปรแกรมคอมพิวเตอร์ที่ต้องการมีความถูกต้องและรวมถึงการตรวจสอบโปรแกรมคอมพิวเตอร์ที่มีการพัฒนามาก่อนแล้ว เนื้อหาในบทนี้จะเป็นการทบทวนความรู้พื้นฐานในการเขียนโปรแกรมคอมพิวเตอร์

เนื้อหา

- ประวัติการเขียนโปรแกรมคอมพิวเตอร์
- หลักการและแนวคิดการเขียนโปรแกรมคอมพิวเตอร์
- ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์
- ผังงานการเขียนโปรแกรมคอมพิวเตอร์

ประวัติการเขียนโปรแกรมคอมพิวเตอร์

First programming languages

- 1951 – Regional Assembly Language
- 1952 – Autocode
- 1954 – IPL (forerunner to LISP)
- 1955 – FLOW-MATIC (led to COBOL)
- 1957 – FORTRAN (First compiler)
- 1957 – COMTRAN (precursor to COBOL)
- 1958 – LISP
- 1958 – ALGOL 58
- 1959 – FACT (forerunner to COBOL)
- 1959 – COBOL
- 1959 – RPG
- 1962 – APL
- 1962 – Simula
- 1962 – SNOBOL
- 1963 – CPL (forerunner to C)
- 1964 – Speakeasy (computational environment)
- 1964 – BASIC
- 1964 – PL/I
- 1966 – JOSS
- 1967 – BCPL (forerunner to C)

https://en.wikipedia.org/wiki/History_of_programming_languages

ประวัติการเขียนโปรแกรมคอมพิวเตอร์

Current trends

- 2000 – ActionScript
- 2001 – C#
- 2001 – D
- 2002 – Scratch
- 2003 – Groovy
- 2003 – Scala
- 2005 – F#
- 2006 – PowerShell
- 2007 – Clojure
- 2009 – Go
- 2010 – Rust
- 2011 – Dart
- 2011 – Kotlin
- 2011 – Red
- 2012 – Julia
- 2014 – Swift
- 2016 – Ring

https://en.wikipedia.org/wiki/History_of_programming_languages

ประวัติการเขียนโปรแกรมคอมพิวเตอร์

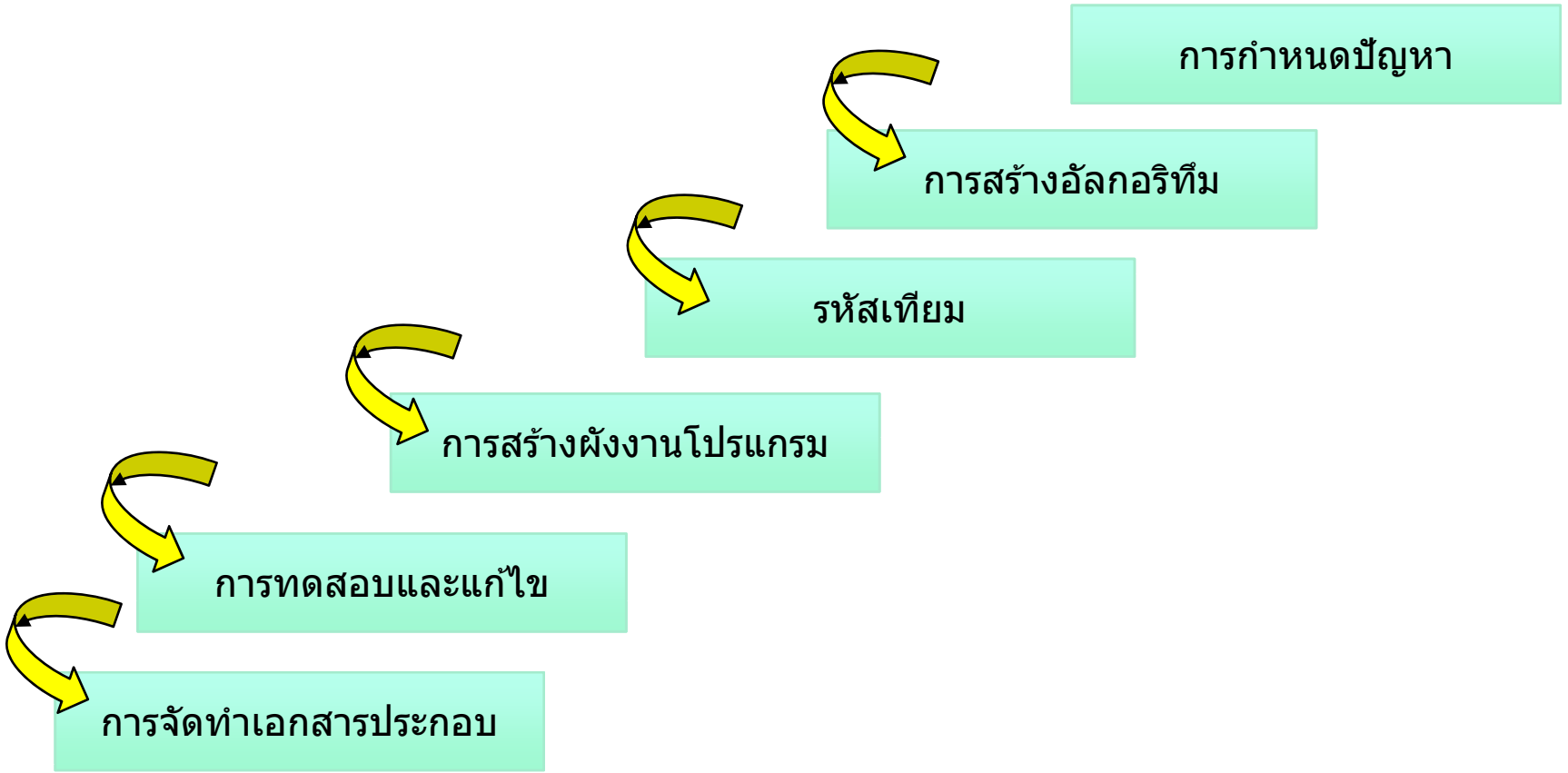
https://i0.wp.com/geoawesomeness.com/wp-content/uploads/2014/08/progLanguages.jpg?resize=581%2C371

ที่มาจากเว็บไซต์ http://sogrady-media.redmonk.com/sogrady/files/2015/07/lang-rank-615-wm.png

หลักการและแนวคิดการเขียนโปรแกรมคอมพิวเตอร์

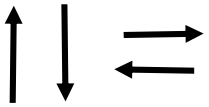
ความต้องการ Requirement	ภาษากลาง Code	โปรแกรม Program
ความต้องการของผู้ใช้งานสำหรับโปรแกรมหรือซอฟท์แวร์ที่จะพัฒนาขึ้น	การพัฒนาโปรแกรมโดยใช้ภาษาทางคอมพิวเตอร์ให้สามารถทำได้ตามความต้องการของผู้ใช้งาน	โปรแกรมที่พัฒนา

ขั้นตอนการเขียนโปรแกรมคอมพิวเตอร์
(Programming Process)



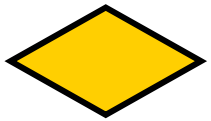
ผังงานการเขียนโปรแกรม

terminal
processing



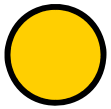
arrow

Input
output



decision

document



on page
connector

off page
connector

display

Unity 3D Concepts

Unity คือ Game Engine ที่ช่วยสร้างเกม 3 มิติ ซึ่งสามารถทำงานได้ บน 2 แพลตฟอร์ม คือ Windows และ OSX และสามารถ Export งานเพื่อนำไปใช้งานได้หลายแพลตฟอร์ม เช่น

-Windows

-OSX

-Androids

-iOS (iPhone)

-WEB

Unity 3D Concepts

Unity เป็นเครื่องมือช่วยสร้างเกมสามมิติ (ข้อแตกต่างระหว่างโลกสองมิติและสามมิติ คือแกน Z หรือความลึกที่เพิ่มเข้ามา พูดง่ายๆก็คือ นอกจากเราจะเคลื่อนที่ขึ้น/ลง บนหน้าจอได้ ยังสามารถเคลื่อนที่ เข้าไปในจอได้)

Unity มองทุกอย่างเป็น GameObject ไม่ว่าจะเป็นก้อนหินก้อนหนึ่ง หรือ แมลงตัวหนึ่ง ถือเป็น GameObject โดย GameObject จะทำงานร่วมกับ Component

Unity 3D Concepts

- **Game Object** คือวัตถุต่างๆที่อยู่ในเกมส์ เช่น รถ 1 คัน,สัตว์ 1 ตัว,คน 1 คน,บ้าน 1 หลัง เป็นต้น
- **Component** คือคุณลักษณะหรือความสามารถต่างๆ ของ Object เช่น การเคลื่อนไหว
- **Asset** คือ คุณลักษณะภายนอกที่เสริมการทำงานของ Component
- **Scence** คือ ฉากแต่ละฉากซึ่งประกอบด้วย Game Object หลายๆ ตัวรวมกัน

Unity 3D Concepts

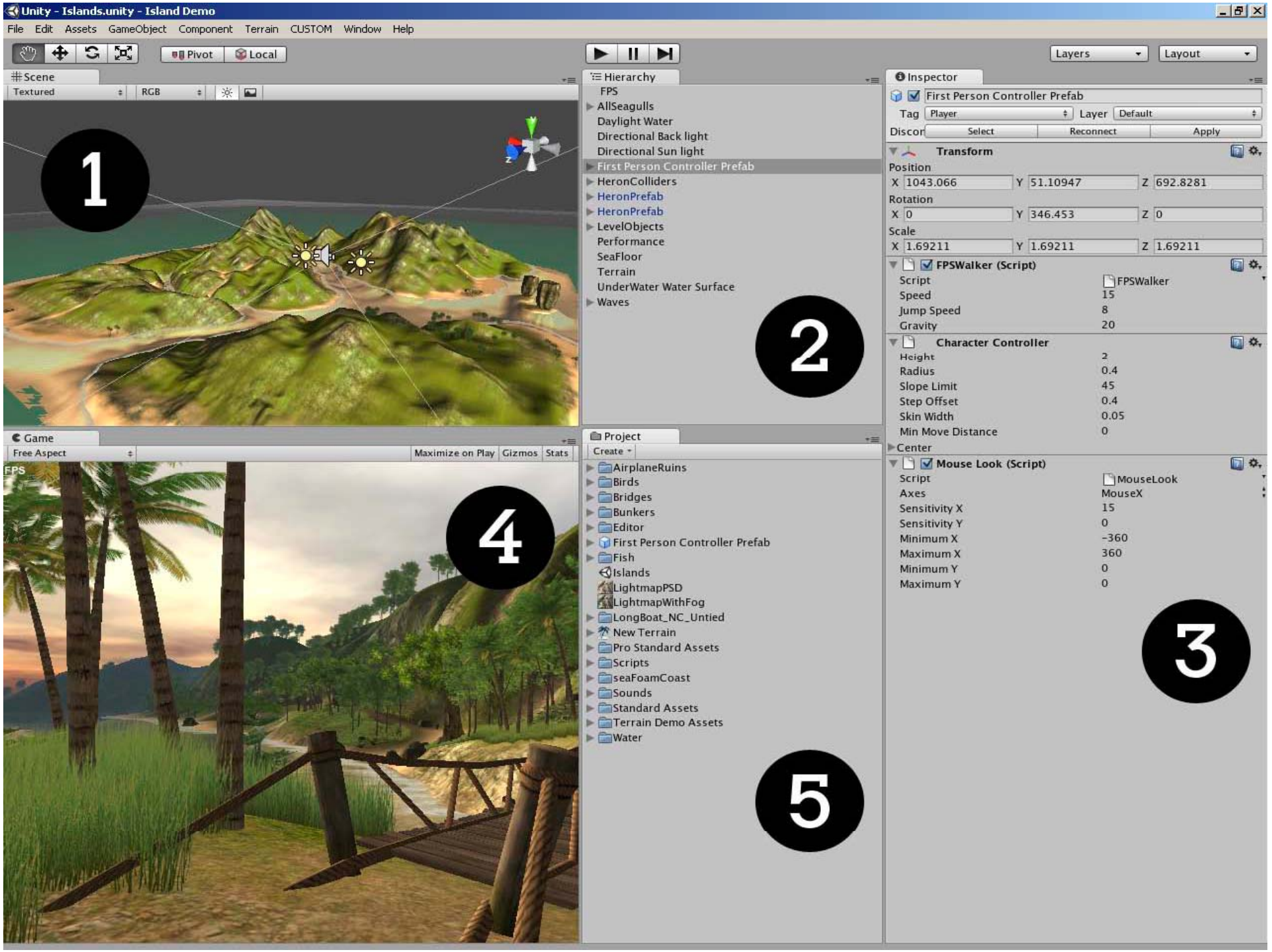
- **Assets** - building blocks of all Unity projects - graphics (textures), models, sound files. The files you use to create the scenario are stored in a folder called Assets
- **Scenes** - scenes are individual levels, areas of game content. Scenes can be loaded on demand.
- **Game Objects** - assets used in the scene become GameObjects (script name) All GameObjects have at least one component - the Transform component.
- **Components** - come in various forms. Attach components to and object to add parts of the game engine to the the component e.g a physics component, or a script component

Unity 3D Concepts

- **Scripts คือ** เป็นการสั่ง components เพิ่ม ขยายหรือเปลี่ยนแปลงคุณลักษณะของ Game Objects ใช้ class ในการกำหนดคุณลักษณะ
- **Prefabs** - เป็นส่วนประกอบที่เกี่ยวข้องกับ Game Objects โดยทำการเก็บ components และ configuration สามารถนำมาใช้ใน scenes หรือ นำไปใช้ในเกมอื่นได้ ตัวอย่างเช่น การควบคุม Controller

Unity 3D Concepts

- **Scripts** - components used to add, extend or modify behavior of game objects. Unity uses a Behavior class to facilitate the use of custom behaviours.
- **Prefabs** - prefabricated game objects with stored associated components and configuration. Prefabs allow functional game objects to be reused in scenes (spawned) or imported into other projects as external assets. ‘The First Person Controller’ is an example of a Prefab


Unity - Islands.unity - Island Demo
File Edit Assets GameObject Component Terrain CUSTOM Window Help
Pivot Local
Layers Layout
Scene
Textured RGB
1
Hierarchy
FPS
AllSeagulls
Daylight Water
Directional Back light
Directional Sun light
First Person Controller Prefab
HeronColliders
HeronPrefab
HeronPrefab
LevelObjects
Performance
SeaFloor
Terrain
UnderWater Water Surface
Waves
2
Inspector
First Person Controller Prefab
Tag Player Layer Default
Discor Select Reconnect Apply
Transform
Position
X 1043.066 Y 51.10947 Z 692.8281
Rotation
X 0 Y 346.453 Z 0
Scale
X 1.69211 Y 1.69211 Z 1.69211
FPSWalker (Script)
Script FPSWalker
Speed 15
Jump Speed 8
Gravity 20
Character Controller
Height 2
Radius 0.4
Slope Limit 45
Step Offset 0.4
Skin Width 0.05
Min Move Distance 0
Center
Mouse Look (Script)
Script MouseLook
Axes MouseX
Sensitivity X 15
Sensitivity Y 0
Minimum X -360
Maximum X 360
Minimum Y 0
Maximum Y 0
3
Game
Free Aspect Maximize on Play Gizmos Stats
FPS
4
Project
Create
AirplaneRuins
Birds
Bridges
Bunkers
Editor
First Person Controller Prefab
Fish
Islands
LightmapPSD
LightmapWithFog
LongBoat_NC_Untied
New Terrain
Pro Standard Assets
Scripts
seaFoamCoast
Sounds
Standard Assets
Terrain Demo Assets
Water
5

Interface

- **Scene** [1]—where the game is constructed
- **Hierarchy** [2]—a list of GameObjects in the scene
- **Inspector** [3]—settings for currently selected asset/object
- **Game** [4]—the preview window, active only in play mode
- **Project** [5]—a list of your project's assets, acts as a library

Interface

- **Scene** [1]—where the game is constructed
- **Hierarchy** [2]—a list of GameObjects in the scene
- **Inspector** [3]—settings for currently selected asset/object
- **Game** [4]—the preview window, active only in play mode
- **Project** [5]—a list of your project's assets, acts as a library