2D Mode & Particles

Change modes between 2D or 3D mode

- Bring up to the Editor Settings Inspector, via the Edit>Project Settings>Editor menu.
- Then set Default Behavior Mode to either 2D or 3D.

In 2D Project Mode:

- Any images you import are assumed to be 2D images (Sprites) and set to Sprite mode.
- The Sprite Packer is enabled.
- The Scene View is set to 2D.
- The default game objects do not have real time, directional light.
- The camera's default position is at 0,0,–10. (It is 0,1,–10 in 3D Mode.)
- The camera is set to be Orthographic. (In 3D Mode it is Perspective.)
- In the Lighting Window:
    - Skybox is disabled for new scenes.
    - Ambient Source is set to Color. (With the color set as a dark grey: RGB: 54, 58, 66.)
    - Precomputed Realtime GI is set to off.
    - Baked GI is set to off.
    - Auto-Building set to off.

Sprites

- Sprites are 2D Graphic objects. If you are used to working in 3D, Sprites are essentially just standard textures but there are special techniques for combining and managing sprite textures for efficiency and convenience during development.
- Sprite Creator - Add Component -> Sprite
- Use the Sprite Creator to create placeholder sprites in your project, so you can carry on with development without having to source or wait for graphics.
- Sprite Editor - In the import settings click on Sprite Editor
- The Sprite Editor lets you extract sprite graphics from a larger image and edit a number of component images within a single texture in your image editor. You could use this, for example, to keep the arms, legs and body of a character as separate elements within one image.

Sprite Renderer

- Sprites are rendered with a Sprite Renderer component rather than the Mesh Renderer used with 3D objects. Use it to display images as Sprites for use in both 2D and 3D scenes.
- Draw Mode - Select an option from the Draw Mode drop-down box to define how the Sprite scales when you change its dimensions.
    - Simple - This is the default Sprite Renderer behavior. The image scales in all directions when its dimensions change.
    - Sliced - Use Sliced if you intend to apply 9-slicing to an image, and you want the sections to stretch. In Sliced mode, the corners stay the same size, the top and bottom of the Sprite stretch horizontally, the sides of the Sprite stretch vertically, and the centre of the Sprite stretches horizontally and vertically to fit the Sprite's size. See documentation on 9-slicing Sprites for more information.
    - Tiled - Use Tiled if you intend to apply 9-slicing to an image, and you want the sections to repeat. In Tiled mode, the sprite stay the same size, and does not scale. Instead, the top and bottom of the Sprite repeat horizontally, the sides repeat vertically, and the centre of the Sprite repeats in a tile formation to fit the Sprite's size. See documentation on 9-slicing Sprites for more information.
- Sorting Layer - The layer used to define this sprite's overlay priority during rendering.
- Order In Layer - The overlay priority of this sprite within its layer. Lower numbers are rendered first and subsequent numbers overlay those below.
- Sprite Mask - Use to apply mask by adding child to the mask gameobject and then choose the MaskInteraction from the CHILD.

2D Physics

- Same as in 3D, but without the Z axis.
- Colliders - just like 3D, but without the Z axis
- 2D RigidBody - pretty much the same as the normal, but without the Z Axis
- Body Types - There are three options for Body Type. Any Collider 2D attached to a Rigidbody 2D inherits the Rigidbody 2D's Body Type. The three options are:
    - Dynamic - Normal mode where physic forces are applied - Gravity and such
    - Kinematic - A Kinematic Rigidbody 2D is designed to move under simulation, but only under very explicit user control. While a Dynamic Rigidbody 2D is affected by gravity and forces, a Kinematic Rigidbody 2D isn't. For this reason, it is fast and has a lower demand on system resources than a Dynamic Rigidbody 2D. Kinematic Rigidbody 2D is designed to be repositioned explicitly via Rigidbody2D.MovePosition or Rigidbody2D.MoveRotation. Use physics queries to detect collisions, and scripts to decide where and how the Rigidbody 2D should move.
    - Static - no need of explanation here :)

- Simulated - Enable Simulated (check the box) if you want the Rigidbody 2D and any attached Collider 2Ds and Joint 2Ds to interact with the physics simulation during run time. If this is disabled (the box is unchecked), these components do not interact with the simulation
- Sleeping Mode - Define how the GameObject "sleeps" to save processor time when it is at rest.
    - Never Sleep - Sleeping is disabled (this should be avoided where possible, as it can impact system resources).
    - Start Awake - GameObject is initially awake.
    - Start Asleep - GameObject is initially asleep but can be woken by collisions.

2D Joins (pretty much the same as 3D)

- Most Common types
    - Spring - Same as spring in 3D. Tries to get to the anchor with spring like behaviour
    - Distance - is a 2D joint that attaches two GameObjects controlled by Rigidbody 2D physics, and keeps them a certain distance apart.
    - Hinge - The Hinge Joint 2D component allows a GameObject controlled by RigidBody 2D physics to be attached to a point in space around which it can rotate.
    - Wheel - Simulates wheel, by adjusting the motor, suspension and anchors you can set up any vehicle. Best way to learn it is by experimenting with the settings.

Particle Systems

- In a 3D game, most characters, props and scenery elements are represented as meshes, while a 2D game uses sprites for these purposes. Meshes and sprites are the ideal way to depict "solid" objects with a well-defined shape. There are other entities in games, however, that are fluid and intangible in nature and consequently difficult to portray using meshes or sprites. For effects like moving liquids, smoke, clouds, flames and magic spells, a different approach to graphics known as particle systems can be used to capture the inherent fluidity and energy. This section explains Unity's particle systems and what they can be used for.