**Không load dc Store: do k login dc -> Bật VPN**

**FRAME\_OUT.. -> Khi run game bị dừng ở splash screen của cocos, k vào game dc -> Update window.**

**FBX (filmbox?) là gì?**

**Forum hỏi đáp: discuss.cocos2d-x.org/c/creator**

**Có cách nào để responsive UI không? Ngoài fit height, fit width? Ví dụ set fit width xong scale cái height để nó fit nè? ->** [**https://docs.cocos.com/creator/manual/en/components/widget.html**](https://docs.cocos.com/creator/manual/en/components/widget.html) **?**

**Tránh copy 1 đống resource vào asset -> McAfee nó nghi ngờ virus.**

**Hiện mấy dòng chữ đỏ khi bật Cocos Editor -> Tắt đi bật lại.**

**Đối vs các animated object thì xét collider kiểu gì?**

* **Thuật toán play animation khi di chuyển player.**
* **Thuật toán tìm vị trí spawn bomb thích hợp.**

1. **Vụ đè đầu**
2. **Animated/Kinematic thì đi xuyên Static, còn Dynamic thì nó đẩy bay đi xa**

* **Để Dynamic thì lâu lâu nó đi xuyên luôn.**

1. **Cách chỉnh property của các key frame sprite? Lúc animation quay ngang bị kẹt cái đầu, k chui khe dc.**
2. **Nhấn Space là bị khựng player.**

**Anh Hùng nói player có rigidbody là dc rồi, các block obstacles chỉ cần có collider thôi.**

**Convention**

[**http://docs.cocos.com/creator/manual/en/scripting/reference/coding-standards.html**](http://docs.cocos.com/creator/manual/en/scripting/reference/coding-standards.html)

[**https://giasutamtaiduc.com/cong-thuc-osascomp.html**](https://giasutamtaiduc.com/cong-thuc-osascomp.html)

* Biến, hàm: camelCase.
* Type, module: PascalCase.
* Hằng: SNAKE\_CASE\_BUT\_ALL\_ARE\_UPPERCASE
* private \_bombAmount: number;
* file: game-controller.ts
* var obj = Object.create(null);
* var array = [];
* exampleString = ‘Better use single quotation mark’;
* Nếu string quá dài, dùng “+”.
* ===, !== thay vì ==, !=.
* howAFunctionLookLike() {

}

* // space before each comment.

**Di chuyển**

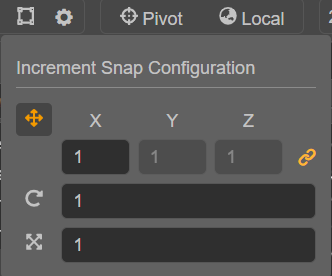
* Giống Unity.

**Phím tắt (Cocos Creator -> Shortcuts)**

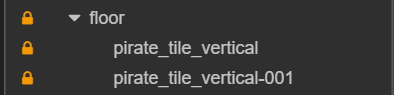
* Đè shift + click item đầu + click item cuối: Chọn 1 dãy node.



* W – Move Gizmo: move.
* E – Rotate Gizmo: rotate.
* R – Scale Gizmo: scale = cách nắm trục kéo.
* T – Rect Gizmo: scale = cách nắm góc kéo giống kéo ảnh.



* Muốn mỗi khi mình sử dụng và drag các gizmo bên trên, offset cố định thì vào đây tick vô gizmo tương ứng. (Bên Unity phím tắt là Ctrl + drag)
* Alt + click: lock cả cha lẫn con



**Mindset**

**Camera**

* Khi mới tạo project nó sẽ cho mình sẵn 1 cái Main Camera trong scene, kệ thằng camera này đi, khi mình kéo sprite vào panel Scene nó sẽ tự sinh ra 1 cái Canvas bên Hierarchy cho mình, trong Canvas này có 1 Camera chỉa vuông góc xuống mặt phẳng canvas -> Sau đó làm game 2D thì cứ tương tác trong cái canvas này thôi.

**Game background**

* Kéo ảnh vào panel Scene để tạo sprite, sau đó scale nó lên (2, 2, 2) để nó fit với cả các device có kích thước màn hình to.

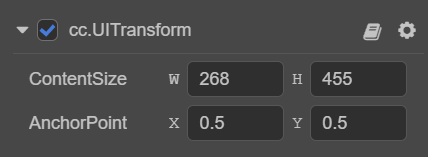
**Things**

**Inspector**

* Mỗi dòng nho nhỏ như này gọi là property



* Mỗi component chứa 1 or nhiều property



* + Cái dấu tick đó cũng là 1 property, gọi là “enabled property”
* Mỗi node (game object) chứa 1 or nhiều component.
  + Dấu tick của node gọi là “activate property”

**Node (tương ứng với game object bên Unity)**

* Là bất kỳ objects nào trong game
  + Vd: Map, player, camera, weapon,..
* Các property cơ bản
  + Position
  + Rotation
  + Scale
  + Layer
* Muốn thêm gì thì Add Component như Unity.

**Script**

**cc.Sprite**

* Là các item có visual component.
  + Vd: background, floor,..
* Muốn tạo sprite, đơn giản nhất cứ kéo mẹ \*.png vào panel Scene.

**Asset**

* Scenes, prefabs, scripts, sprites, sound,..

**Material**

* Dùng để làm “Skin” của object (node).
* Muốn add material vào node thì add component MeshRenderer vào node, xong kéo material vào “Materials” của component này.
* Nếu node k hiển thị skin tức là bị mất material, xem [link](https://www.youtube.com/watch?v=Am1fcw9LQPY&list=PLbvpmJKjO3NA4dlW43GzhJUMaXyIp3xpc&index=4&ab_channel=Cocos).

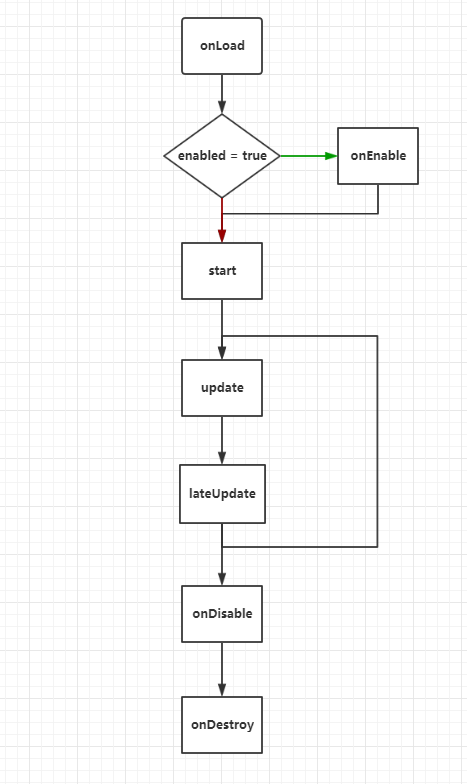
**Texture**

* 1 tấm ảnh để dán lên material.
  + Click vào material.
  + Tick ô USE ALBEDO…
  + Kéo texture vào.

**Prefab**

* Giống Unity và có dạng 3 hộp vuông xếp chồng nhau.
* 2 cái ngoài cùng
* 
  + Reset from prefab: reset các thông số của node về thông số gốc của prefab
  + Update prefab asset: apply các thông số của node lên prefab -> các anh em của nó cũng sẽ bị đổi theo.

**Life cycle callback –** [**link**](https://docs.cocos.com/creator/3.3/manual/en/scripting/life-cycle-callbacks.html)



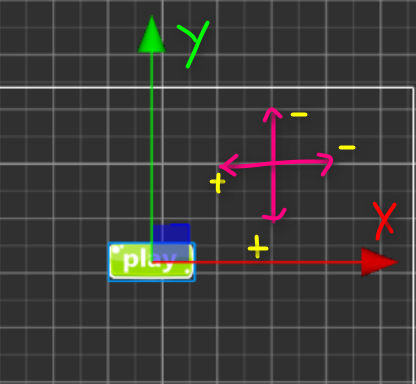
* onLoad(): execute đầu tiên khi scene được load lên or node dc activate.
* onEnable(): dc gọi khi component được enable.
* onDisable(): dc gọi khi component bị disable.
* start(): execute sau onEnable, dc gọi DUY NHẤT 1 LẦN khi component được khởi tạo thành 1 instance.
* update(): execute ở mỗi frame (tần suất execute khác nhau ở mỗi máy, tùy thuộc vào tốc độ render của máy).
* lateUpdate(): giống update, execute sau update.
* onDestroy(): dc gọi khi component bị destroy.

**Light**

* Directional light:
  + Illuminance – độ rọi: số càng to, chiếu càng sáng.
* Sphere light:
  + Size: bán kính của hình cầu (màu trắng) có tâm là nguồn sáng.
  + Range: bán kính của hình cầu (màu vàng) có tâm là nguồn sáng, là khoảng cách tối đa mà ánh sáng chiếu tới.
* Điều chỉnh 2 cái này để ra dc Luminous flux (tổng lượng ánh sáng phát ra từ một nguồn sáng) phù hợp (nhìn vừa mắt là dc)
* Spotlight: giống sphere light nhưng có thêm angle
  + Angle: độ rộng của chum tia từ đèn pin.

**cc.UITransform**

* AnchorPoint(x, y) – điểm neo: với x có đơn vị là % chiều rộng của node, y có đơn vị là % chiều dài của node
  + Vd: Mặc định AnchorPoint(0.5, 0.5) – tức trục Oxy đứng yên, node di chuyển qua trái 50% chiều rộng, 50% chiều dài -> Khi này gốc tọa độ O nằm tại tâm của node.



* + Mặc định các gizmo move, rotate,.. ở chế độ Pivot- tức lấy tâm là anchor point của node. Nếu muốn lấy tâm dịch chuyển là trọng tâm của node -> Chỉnh qua Center.



**cc.Camera**

* Muốn camera view nó fit với canvas thì chỉnh OrthoHeight (chiều cao khung hình ở chế độ trực giao) = height của canvas.

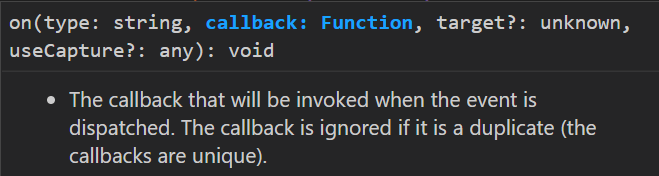
**Script (Code) – Shift + Alt + F để làm đẹp code**

**@property({type: …})**

* Dùng để serialize object – tức lưu object này vào đâu đó để sau này lấy ra xài lại.
* Nói trắng ra là để cái biến đó nó xuất hiện trong Cocos Creator để mình chỉnh giá trị trực tiếp luôn (như public class member bên Unity vậy).

**this.node**

* Gọi ra dc các attributes cơ bản của node: .position, .angle (rotation), .scale, .layer.
* this.node.parent: lấy parent của nó trong Hierarchy.
* This.node.on



**Node System Events – mouse, touch**

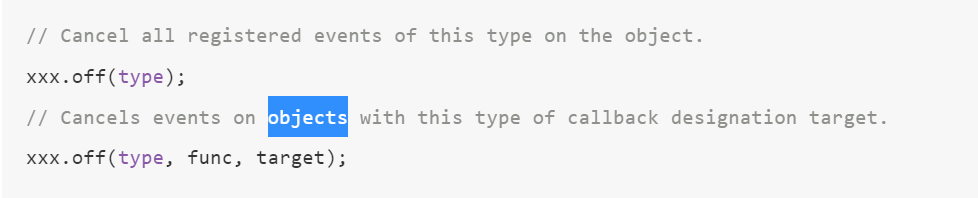
* Đọc từ docs ngắn gọn, dễ hiểu.
* This.node.on(type of event, callbackFunction(), ?target)
  + Type of event: Dùng string dc, nhưng tốt nhất nên dùng enumeration.
  + callbackFunction(): hàm được gọi ghi event bị trigger.
  + Target: nếu trong callbackFunction() có dùng tới syntax “this”, thì truyền object vào để hàm callback kia nó biết nên gọi tới attributes của object nào. Nếu k truyền, mặc định nó trỏ tới object đang execute hàm callback.

this.node.on(Node.EventType.MOUSE\_DOWN, function ( event ) {

this.enabled = false;

}.bind(this));

* Khỏi dùng bind cũng dc, lí do thì ở trên nói rồi.
* ***Nhớ cancel khi không xài nữa (hoặc dùng once thay cho on thì khỏi cancle):***



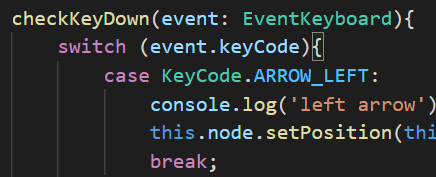
* **DKY Ở onEnable(), CANCEL Ở onDisable()? – trong doc example nó làm vậy.**

**Global System Events – mouse, touch, keyboard, device motion**

* Giống ở trên, nhưng thay “this.node” = “systemEvent”
* **DKY Ở onLoad(), CANCEL Ở onDestroy()? – trong doc example nó làm vậy.**
* Keyboard event:

systemEvent.on(SystemEvent.EventType.KEY\_DOWN, this.checkKeyDown, this);

* + Sau khi dky ở onLoad(), implement hàm checkKeyDown để xem người ta gõ phím gì -> thực hiện hành động mình muốn.
  + checkKeyDown:



**Scene**

* director.preloadScene(“Scene1”, function() {…}): nạp scene trong thầm lặng ở background.
* director.loadScene(“Scene1”): load scene xong sau đó chuyển game sang scene mới (chứ k nhất thiết phải gọi preloadScene).
* 2 thằng này nằm khác script thì vẫn chạy dc bth.

**Pause/Resume game**

* Trong docs version 3.3 k thấy có game.pause()/.resume()

**Sound**

**Animation (**[**link**](https://docs.cocos.com/creator/3.3/manual/en/editor/animation/animation-editor.html) **này bấm từ panel animation ra, kiếm trong docs.cocos.com k có)**

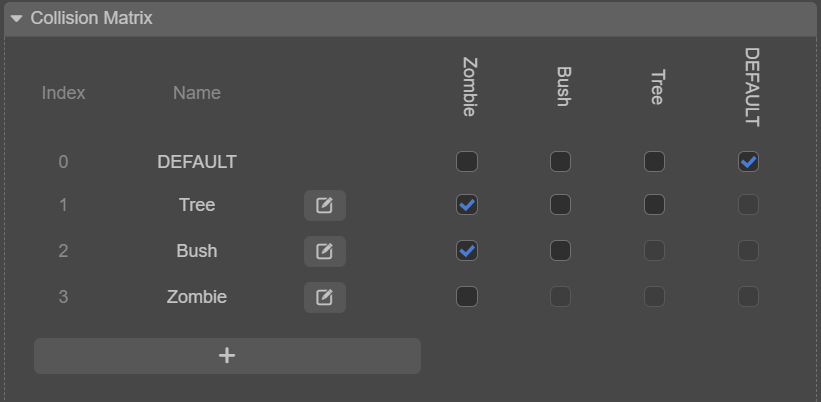
* Khái niệm: animation ở đây dc hiểu là đoạn phim quay lại sự thay đổi TRÊN TỪNG PROPERTY. Sự thay đổi này được biểu thị qua các cột mốc (trạng thái) gọi là key frame. Animation có thể trình chiếu 1 property, hoặc nhiều property cùng lúc.
  + Vd: Muốn tạo animation character walking -> Cần quay lại sự thay đổi về tư thế của sprite Character (property cc.spriteFrame) -> Cần 1 chuỗi các ảnh png nối liền nhau để miêu tả hành động -> Mỗi ảnh png là 1 cột mốc (key frame).
    - ***Nếu đống png có size giống nhau mà khi vào key frame có vài cái giãn ra thì cc.Sprite -> SizeMode = TRIMMED ~~CUSTOM~~.***
      * ***Lí do: nếu nhìn kĩ các png sẽ có max width và max height khác nhau (ví dụ character đang đưa tay ra -> width rộng hơn), nếu để CUSTOM nó sẽ auto scale tất cả png sao cho khớp với 1 size duy nhất.***
* Cơ bản:
  + Add Component “Animation” vào node.
  + Bấm dấu + bên assets, tạo file animation clip.
  + Kéo file vừa tạo vào phần clip của component kia.
* Trong Property List, data của mỗi property chính là đám key frames trên Timeline.
  + Các property có thể add vào list để tạo animation gồm
    - 4 property cơ bản của node.
    - Các property của tất cả components mà cái node đang chứa cc.Animation này sở hữu.
* Di chuyển:
  + Double click vào key frame để thanh timeline move tới đó.
  + Nhấn vào chỗ trống rồi kéo lưới để chọn nhiều key frames.
* Tạo key frame:
  + Double click vào vị trí muốn add trên timeline cho cây kim time nó nhảy tới đó.
  + Thay đổi giá trị của property mà mình đang record, khi này 1 key frame mới sẽ dc add. Or nhấn vào thanh này rồi tìm giá trị mình cần Or kéo trực tiếp asset vào.



* Scripting:
  + this.getComponent(Animation).play();
  + Cái này chỉ hoạt động khi đặt từ onEnable trở xuống.
  + Methods: play(), crossFade(), pause(), resume(), stop().

**cc.{Box/Circle/Pollygon}Collider2D**

* Viền xanh lá.
* Box: viền hình vuông.
* Circle: viền hình tròn.
* Pollygon: viền bao sát sprite.
* Project Settings -> Physics -> Tạo ma trận để allow thằng nào dc collide với thằng nào -> Quay lại Inspector -> Chọn group tương ứng cho thằng cc.Collider2D



**Physics Manager(2D)**

* Muốn thằng này works thì node đó phải có cc.Rigidbody2D và cc.Collider2D

PhysicsSystem2D.instance.debugDrawFlags = EPhysics2DDrawFlags.All;

**RigidBody**

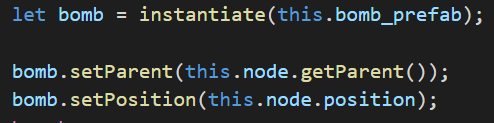
* **Đối với rigidbody, chỉ có rotate hoặc position thôi, không có scale.**
* Để ý vụ scale trong doc nó nói.
* 4 loại body:
  + Static: m = 0, v = 0, k bị ảnh hưởng bởi lực hoặc trọng lực, có thể set position để move nó.
  + Dynamic: có m, có v, có bị ảnh hưởng bởi trọng lực (và lực?).
  + Kinematic: m = 0, v tùy ý, k bị ảnh hưởng bởi trọng lực, move = cách set v or set position.
  + Animated: kế thừa từ kinematic, dùng cho những thằng rigidbody có animation để tránh nó đi xuyên vật thể.
* FixedRotation: Để khi va chạm, object của mình k bị xoay ngang xoay dọc.

**Contact callback (các hàm dc trigger khi 2 rigidbody va chạm)**

* Đọc doc nó chi tiết và dễ hiểu hơn.

**Tạo node/Hủy node từ script**

* Tạo



* Hủy

