دست نویس های کلاس های یونیتی:

1. در دوربین اصلی، میشه رنگ پس زمینه را عوض کرد. در حالت پیشفرض برای Clear Flags مقدار Sky Box است.
2. بهترین شیوه برای حرکت بازیکن، استفاده از متد Translate در کلاس transform است.

(Translate (Vector3\*speed\*Time.deltatime

1. متد GetAxis مقدار ورودی کاربر را بصورت داده ی float از صفر تا یک برمیگردند اما متد GetKey تنها مقدار بولین فشرده شدن را بر میگرداند.
2. بهترین راه اضافه کردن prefabs به کد، استفاده از فیلد [SerializedField] است.
3. متد Instantiate در کلاس MonoBehavior =>

Instantiate (prefabs<gameObject>,position<Vector3>,Quaternion.identity (as a rotation))

1. متد Destroy که با دریافت یک GameObject آنرا نابود میکند.
2. Time.time : مقدار زمان سپری شده از کلاس (خوب برای تعیین cool down)
3. می توانید تغییر را در prefabs برای همه اعمال کنید. فقط باید در نمای Inspector prefabs گزینه های overrides را انتخاب کنید.
4. متد Random : مقدار float برمیگرداند. RandomRange(10.0f,20.0f)
5. یادت باشه که گزینه ی trigger را برای شی هایی که با هم برخورد ندارن بگذاری.
6. اضافه کردن tag: inspector view => tag => add tag
7. شیوه دسترسی به تگ یک شی : other.tag
8. گرفتن یک کامپوننت : gameObject.GetComponent<component-type>()
9. شناسایی و پیدا کردن برخورد (collition) بر عهده ی کامپوننت Rigitbody است. این کامپوننت است که اینترفیس های OnTrigger و OnCollition را دارد.
10. شیوه ی استفاده از coroutine : ساخت یک تابع با مقدار بازگشتی IEnumerator . تعریف تابع و برای مقدار بازگشت از سینتکس yield return new wait… استفاده میکنیم. به این صورت تابع از جایی که صدا زده شده بقیه کار را انجام میدهد و بعد از گشت مدتی که برگشت داده شده، به تابع برمیگردد و بقیه کار را انجام میدهد.

نکته: به StartCoroutin تابع صدا زده میشود و شروع میشود.

1. از متد Find در کلاس GameObject میتوانیم بقیه ی آبجت ها را پیدا کرد.
2. تعیین parent برای یک شی: GameObject.trandform.parents
3. در کامپویننت Sprite Rendering شما میتوانید لایه تعریف کنید و همچنین به لایه ها مقدار بدهید. اگر اسم لایه بالاتر از دیگری باشد، آن زوتر نمایش داده میشود (انگار که روی تصویر زیری است) . همچنین اگر عدد لایه بالاتر باشد، همین واکنش را در پی دارد.
4. تب Animation که از windows بدست میآید میتوان کد ها
5. متد Find را میتوان در هر آبجکتی صدا زد اما اگر میخواهی که فرزند یک شی خاصی را داشته باشی، باید بصورت زیر عمل کنی:  
   game-Object-Name.transform.Find(‘child-name’);  
   در واقع GameObject.Find انگار تمامی فرزندان شی را بررسی میکند. اگر دو تا هم اسم بودن، نزدیک ترین را انتخاب میکند.
6. UI ها را در یک شی خالی به اسم UI\_Manager قرار بده
7. در Canvas و در کامپوننت Canvas Scaler، گزینه ی UI Scale Mode را بر روی Scale With Screen Size قرار بده تا با تغییر رزولوشن، اندازه متن ثابت بماند.
8. برای دسترسی به UI در کد، لازم است کتابخانه UI از بسته UnityEngine صدا زده شود.
9. برای تغییر فونت در TEXTMESHPRO باید فونت را به یونتی معرفی کنیم (بسازیم) باری این کار در تب Window بخش TextMeshPro گزینه ی Font Asset Creator را انتخاب و فونت را به یونیتی معرفی میکنیم.
10. برای استفاده از TMP باید از کتابخانه TXpro استفاده کرد
11. از متغیر با تابع TextMeshUGUI زمانی استفاده کن که میخواهی از این کامپوننت در UI استفاده کنی و از TextMeshPro زمانی استفاده کن که در بازی میخواهی از آن استفاده کنی
12. اگر بخوای به اسکریپت بخصوص آبجکتی وصل شی (مثلا دو Enemy که هر دو یک کد دارند اما یکی باید امتیازش کمتر شود) اول باید با Find یا FindWithTags آن را پیدا و سپس کامپوننت را صدا بزنی
13. وقتی image اضافه میشود، باید گزینه ی Preserve Aspect در کامپوننت Image فعال شود تا با تغییر رزولوشن، تصویر بهم نریزد.
14. بهتره برای شی هایی که تاثیری در بازی ندارند، بقیه ی شی ها را با استفاده از فیلد SerializedField بگیرند .(مثل UI\_Manager)
15. میشود با متد SetActive در game Objects ها ، آنها را اکتیو و دی اکتیو کرد
16. در کلاس SceneManager در کتابخانه ی SceneManagement ، متدی است به نام LoadScene که میتواند سکانسی را لود کند. ورودی تابع اسم سکانس یا ایندکس آن است. همچنین میتوان به عنوان تابع دوم به آن حالت شروع سکانس را گفت. برای اینکه سکانسی را به یونیتی معرفی کنیم باید : File => build Setting (ctrl+Shift+B) => Add Open Scenes
17. نکته مهم: اگر شی ای در بازی inactive بود، توسط GameObject.Find پیدا نمیشود. تنها راه استفاده از تگ SerializableField است و دادن شی با Inspector.
18. حتما از UI برای کارهای گرافیکی استفاده کن نه Sprite
19. Post Processing : با مجموعه‌ای از افکت‌ها و فیلترهای تصویر ارائه می‌شود که می‌توانید برای بهبود تصاویر بصری بازی‌های خود، روی دوربین‌های خود اعمال کنید. این پکیج را باید از package manager یونیتی دریافت کرد. (Windows=>Packet Manager)
20. شیوه ی کار با Post Processing: یک شی خالی ایجاد کرده و اسم آن را Post Processing Volume میگذاریم. سپس کامپوننت Post Processing Volume را اضافه میکنیم. اگر میخواهیم که این افکت ها به کل بازی الحاق شود، تیک Globals را میزنیم (معمولا فعال است). دو عنصر Blend Distance و Weight برای لایه بندی بین Post Pricessing ها و وزن آنها استفاده میشود. سپس در پروفایل New را انتخاب کرده، سپس میتوان افکت ها را روی بازی قرار داد.
21. حتما برای فعال شدن این افکت ها لازم است که Post Processing Volume در لایه ای مخصوص باشد و با اضافه کردن Post Processing Layer به دوربین، لایه ی مشخص شده را به دوربین معرفی میکنیم (لایه ها در بالا سمت راست Inspector View میباشند)
22. برای اضافه کردن صدا به بازی، باید از کامپوننت Audio Source استفاده کنیم. این کامپوننت فایل صدا را به عنوان AudioClip میگیرد و به اولین کامپوننت AudioListener میدهد. این کامپوننت مثل میکروفون کار میکند و هر چیزی که میگیرد را پخش میکند.
23. در بخش Output کامپوننت Audio Source میتوانید به کامپوننت Audio Mixer اشاره کنید که تنظیمات میکس را بر روی صدای بازی انجام میدهد.
24. متد play(float delay) دارای دو مدل با یک و صفر ورودی است که جزو کلاس AudioSource است. با استفاده از این تابع میتوان صدا را اجرا کرد.
25. روشی هست که میتوان بدون وصل کردن کامپوننت AudioSource و فقط با پاس دادن فایل صوتی در Audio Clip صدایی اجرا کرد. AudioSource.PlayClipAPoint(clip, posi)

