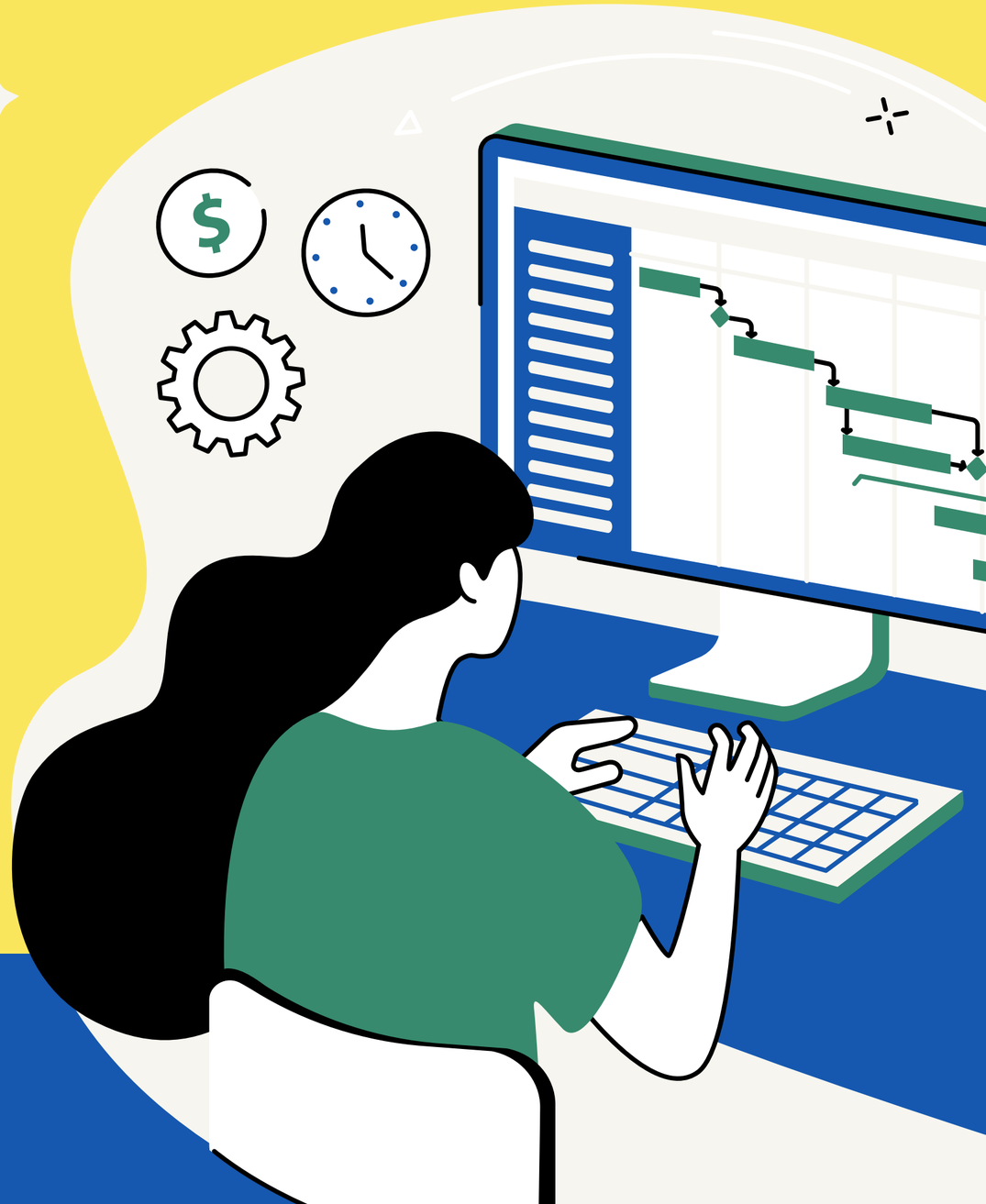
SPĒLES OBJEKTI UNITY DZINĪ

Mācību materiāls

Autore Katrīna Blažkova



KAS IR UNITY GAMEOBJECT?

GameObject ir Unity ainu veidošanas bloki un darbojas kā funkcionālo komponentu konteiners, kas nosaka GameObject izskatu un GameObject darbību.

GameObject pārstāv visu, kas var pastāvēt ainā

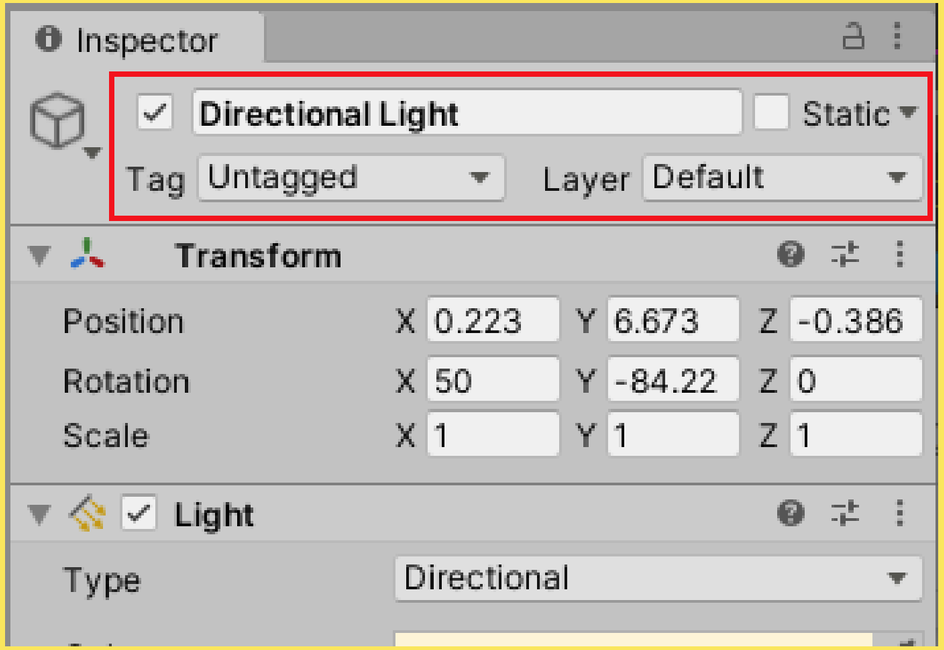
KAS IR UNITY GAMEOBJECT SKRIPTOŠANĀ

Skriptošanā klase GameObject nodrošina metožu apkopojumu, kas ļauj ar tām strādāt savā kodā, tostarp atrast, veidot savienojumus un sūtīt ziņojumus starp GameObjects, kā arī pievienot vai noņemt komponentus, kas pievienoti objektam GameObject, un iestatīt vērtības, kas attiecas uz to statusu notikuma vietā.

AINAS STATUSA REKVIZĪTI

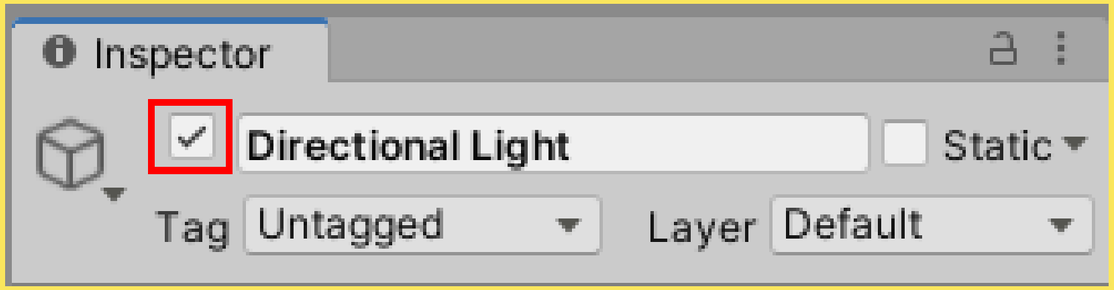
Var izmantot **skriptus** - lai modificētu daudzus rekvizītus, kas saistīti ar GameObject statusu notikuma vietā. Tie parasti atbilst kontrolierīcēm, kas redzamas **inspektora** augšdaļā ja redaktorā ir atlasīts GameObject.

Tie neattiecas ne uz vienu konkrētu komponentu, un tie ir redzami GameObject inspektorā augšdaļā virs komponentu saraksta.



AKTĪVAIS STATUSS (ACTIVE STATUS)

GameObject ir **aktīvs** pēc noklusējuma, bet to var **deaktivizēt**, tādējādi izslēdzot visus GameObject pievienotos komponentus. Tas parasti nozīmē, ka tas kļūs neredzams un nesaņems nevienu no parastajiem callbacks vai notikumiem, piemēram, Update vai FixedUpdate.

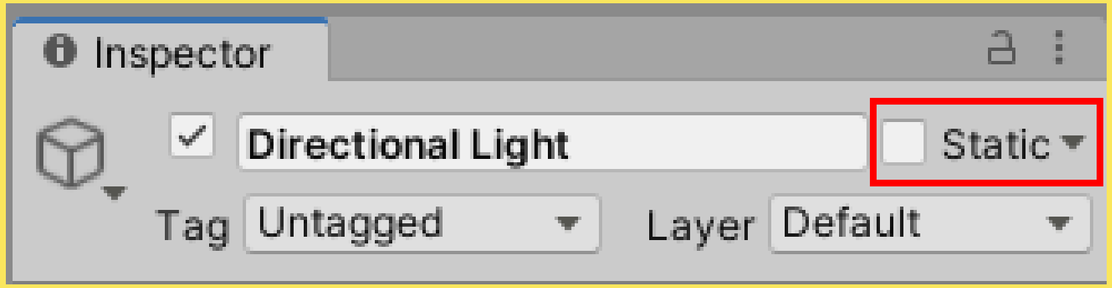


AKTĪVAIS STATUSS (ACTIVE STATUS)

GameObject.activeSelf var izmantot arī, lai nolasītu GameObject pašreizējo aktīvo stāvokli. Izmantojiet **GameObject.activeInHierarchy**, lai nolasītu, vai GameObject tiešām ir aktīvs notikuma vietā.
Gameobject.activeInHierarchy ir nepieciešama, jo to, vai GameObject ir faktiski aktīvs, nosaka tā aktīvais stāvoklis un visu vecāku(parent) aktīvais stāvoklis. Ja kāds no vecākiem nav aktīvs, tad tas nav aktīvs, neskatoties uz savu aktīvo uzstādījumu.

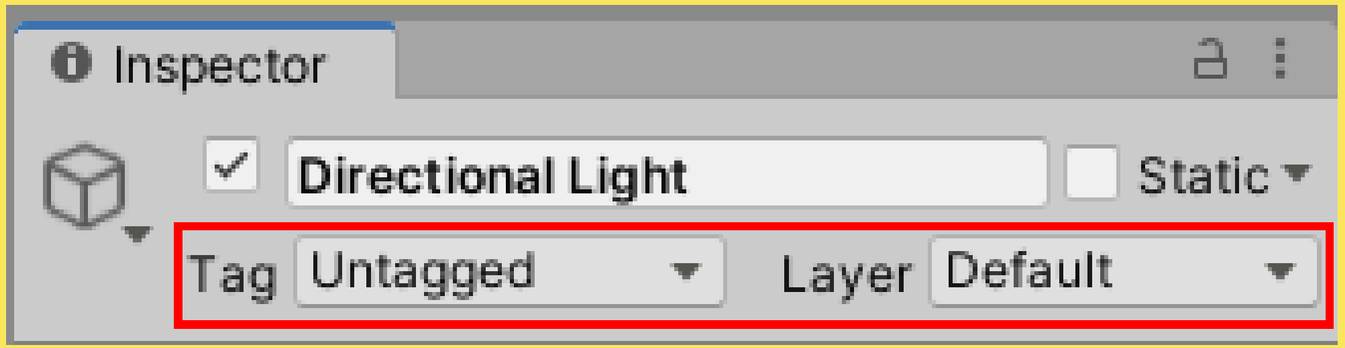
STATIC STATUS (STATIC STATUS)

Dažas Unity sistēmas, piemēram, **Global Illumination**, Occlusion, Batching, Navigation, and **Reflection Probes**, darbojas, balstoties uz GameObject statisko statusu. Jūs varat kontrolēt, kuras no Unity sistēmām uzskata GameObject par statisku, izmantojot **GameObjectUtility.SetStaticEditorFlags**.



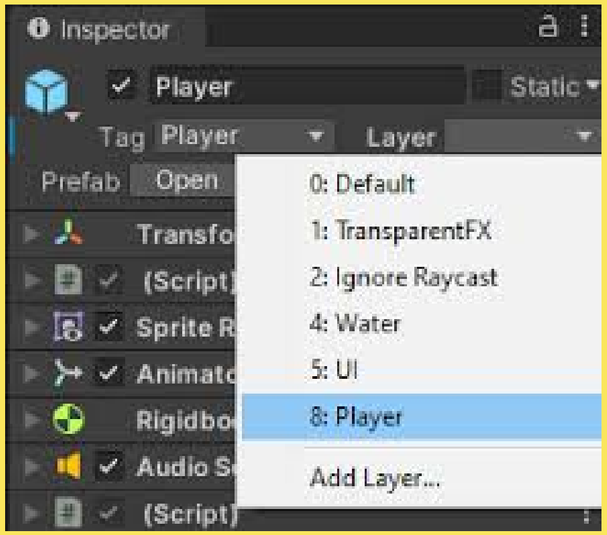
BIRKAS UN SLĀŅI (TAGS AND LAYERS)

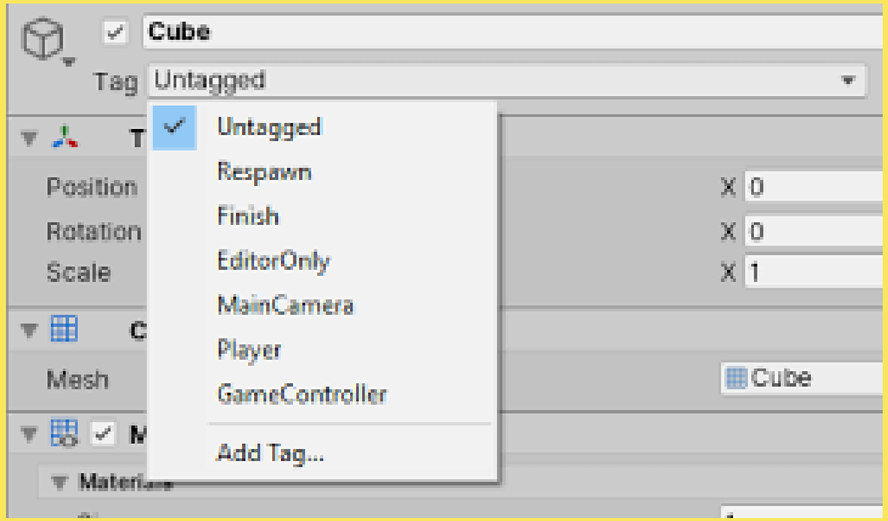
Tagi nodrošina veidu, kā atzīmēt un identificēt dažādus GameObject tipus jūsu ainā, savukārt **Slāņi** piedāvā līdzīgu, bet atšķirīgu veidu, kā iekļaut vai izslēgt GameObject grupas no noteiktām iebūvētām darbībām, piemēram, renderēšanas vai fizikas sadursmēm.



TAGS AND LAYERS

Tagu un slāņu vērtības var modificēt skriptā, izmantojot rekvizītus **GameObject.tag** un **GameObject.layer**. GameObject tagu var pārbaudīt arī efektīvi, izmantojot metodi **CompareTag**, kas ietver taga esamības pārbaudi un neizraisa nekādu atmiņas piešķiršanu.





KOMPONENTU PIEVIENOŠANA UN NOŅEMŠANA

Jūs varat **pievienot** vai **noņemt** komponentus izpildlaikā, kas var būt noderīgi, veidojot GameObject objektus procedurāli vai mainot, kā GameObject uzvedas. Ņemot vērā, ka var arī **iespējot** vai **atspējot** skriptu komponentus un dažus iebūvēto komponentu veidus, izmantojot skriptu, tos neiznīcinot.

Labākais veids, kā pievienot komponentu izpildlaikā, ir izmantot **AddComponent<Type>**, norādot komponenta tipu leņķa iekavās, kā parādīts. Lai noņemtu komponentu, jums jāizmanto **Object.Destroy** metode uz paša komponenta.

PIEKĻŪŠANA KOMPONENTIEM

Vienkāršākais gadījums ir, kad skriptam uz GameObject ir nepieciešams piekļūt citam komponentam, kas ir pievienots tam pašam GameObject (atcerieties, ka citi skripti, kas pievienoti GameObject, arī paši ir komponenti). Lai to izdarītu, pirmais solis ir iegūt atsauci uz komponenta instanci, ar kuru vēlaties strādāt. Tas tiek darīts ar **GetComponent** metodi. Šajā piemērā skripts iegūst atsauci uz **Rigidbody** komponentu tajā pašā GameObject:

```
void Start ()
{
    Rigidbody rb = GetComponent<Rigidbody>();
}
```

PIEKĻŪŠANA KOMPONENTIEM

Kad jums ir atsauce uz komponenta instanci, jūs varat iestatīt tās īpašību vērtības tāpat kā to darītu inspektorā:

```
void Start ()
{
    Rigidbody rb = GetComponent<Rigidbody>();

    // Change the mass of the object's Rigidbody.
    rb.mass = 10f;
}
```

Jūs varat arī izsaukt metodes uz komponenta atsauces, piemēram:

```
void Start ()
{
    Rigidbody rb = GetComponent<Rigidbody>();

    // Add a force to the Rigidbody.
    rb.AddForce(Vector3.up * 10f);
}
```

PIEKĻŪŠANA KOMPONENTIEM

Piezīme: jūs varat pievienot vairākus pielāgotus skriptus vienam un tam pašam GameObject. Ja jums nepieciešams piekļūt vienam skriptam no cita, varat izmantot **GetComponent** kā parasti un vienkārši izmantot skripta klases nosaukumu (vai faila nosaukumu), lai norādītu komponenta tipu, kuru vēlaties. Ja mēģināsiet iegūt komponenta tipu, kas nav pievienots GameObject, **GetComponent** atgriezīs **null**; jūs saņemsiet nulles atsauces kļūdu izpildlaikā, ja mēģināsiet mainīt vērtības uz nulles objekta.

PIEKĻŪŠANA KOMPONENTIEM CITOS GAMEOBJECTS

Lai gan skripti dažkārt darbojas izolēti, bieži ir nepieciešams tiem sekot līdzi citiem GameObject objektiem vai, biežāk, komponentiem uz citiem GameObject objektiem.

Piemēram, gatavošanas spēlē pavāram var būt nepieciešams zināt plīts atrašanās vietu. Unity nodrošina vairākus dažādus veidus, kā iegūt citus objektus, katrs piemērots noteiktām situācijām.

GAMEOBJECTS SAISTĪŠANA AR MAINĪGAJIEM INSPEKTORĀ

Visvienkāršākais veids, kā atrast saistītu GameObject, ir pievienot publisku GameObject mainīgo skriptam:

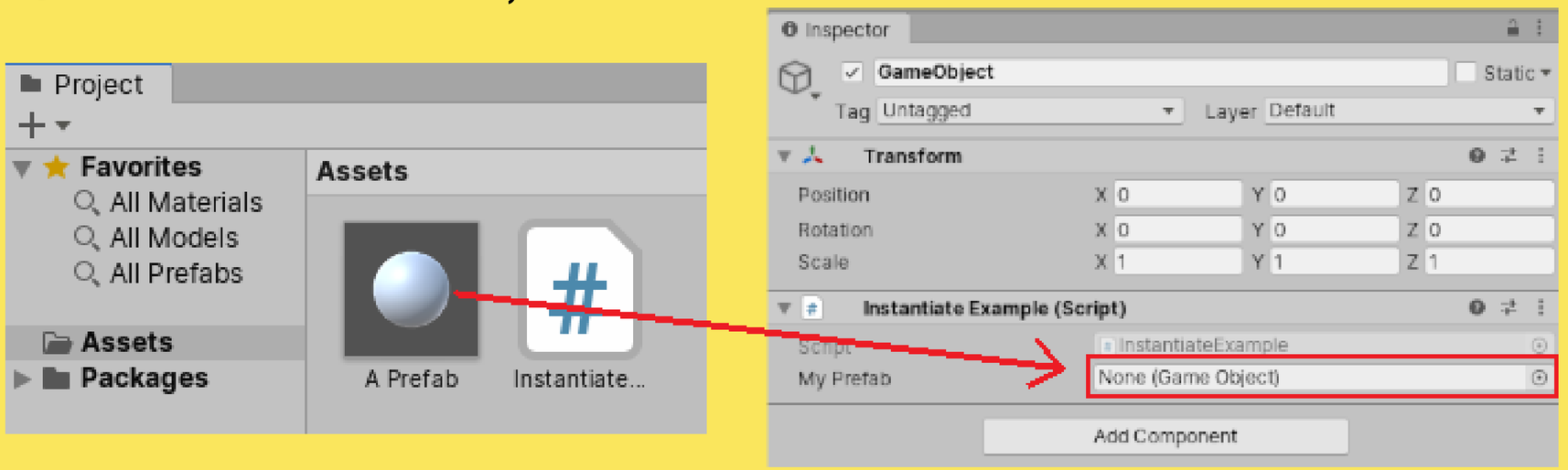
```
public class Chef : MonoBehaviour
{
    public GameObject stove;

    // Other variables and functions...
}
```

Šis mainīgais būs redzams Inspektora panelī kā GameObject field.

GAMEOBJECTS SAISTĪŠANA AR MAINĪGAJIEM INSPEKTORĀ

Tagad jūs varat vilkt objektu no ainavas vai Hierarhijas paneļa uz šo mainīgo, lai to piešķirtu.



Prefabu vilkšana no Projekta loga uz GameObject lauku(field) Inspektora logā

GAMEOBJECTS SAISTĪŠANA AR MAINĪGAJIEM INSPEKTORĀ

GetComponent funkcija un Komponenta piekļuves mainīgie ir pieejami šim objektam tāpat kā jebkuram citam, tāpēc varat izmantot kodu līdzīgi kā zemāk:

```
public class Chef : MonoBehaviour {

    public GameObject stove;

    void Start() {
        // Start the chef 2 units in front of the stove.
        transform.position = stove.transform.position + Vector3.forward * 2f;
    }
}
```

CHILD GAMEOBJECTS ATRAŠANA

Ja spēles ainā ir vairāki GameObjects ar vienādu tipu, piemēram, savācami priekšmeti, ceļa punkti un šķēršļi, tos varētu būt noderīgi uzraudzīt ar konkrētu skriptu (piemēram, visi ceļa punkti var būt nepieciešami ceļa meklēšanas skriptam). Mainīgo izmantošana, lai saistītu šos GameObjects, var padarīt dizaina procesu ķēpīgu, ja katrs jaunais ceļa punkts ir jāvelk uz mainīgo skriptā. Tāpat, ja ceļa punkts tiek dzēsts, ir neērti izņemt mainīgo atsauci uz trūkstošo GameObjectu. Šādos gadījumos bieži vien ir labāk pārvaldīt grupu GameObjects, padarot tos par viena vecāka GameObject bērniem. Bērnu GameObjectus var izgūt, izmantojot vecāka Transform komponentu, jo visiem GameObjectiem ir nepieciešams Transform impliciti.

CHILD GAMEOBJECTS ATRAŠANA

```
using UnityEngine;

public class WaypointManager : MonoBehaviour {
    public Transform[] waypoints;

    void Start()
    {
        waypoints = new Transform[transform.childCount];
        int i = 0;

        foreach (Transform t in transform)
        {
            waypoints[i++] = t;
        }
    }
}
```

Jūs arī varat atrast konkrētu bērna objektu pēc nosaukuma, izmantojot **Transform.Find** metodi: transform.Find("Frying Pan");

ZIŅOJUMU SŪTĪŠANA UN APRAIDE

Rediģējot savu projektu, jūs varat iestatīt atsauces starp GameObjects Inspektora paneli. Tomēr dažreiz ir neiespējami iepriekš iestatīt šos (piemēram, atrast tuvāko objektu personāžam jūsu spēlē vai izveidot atsauces uz GameObjects, kas tika instancēti pēc Ainas ielādes). Šajos gadījumos jūs varat atrast atsauces un nosūtīt ziņojumus starp GameObjects izpildlaiku.

ZIŅOJUMU SŪTĪŠANA UN APRAIDE

BroadcastMessage ļauj jums nosūtīt zvanu uz nosauktu metodi, neesot konkrētam, kur šī metode jāīsteno. Jūs varat to izmantot, lai izsauktu nosauktu metodi uz katru MonoBehaviour uz konkrētu GameObject vai jebkuru no tā bērniem. Jūs varat izvēlēties arī obligāti prasīt, lai vismaz viens saņēmējs būtu (vai tiks ģenerēta kļūda).

SendMessage ir nedaudz specifiskāka un nosūta zvanu uz nosauktu metodi tikai uz paša GameObject un nevis uz tā bērniem.

SendMessageUpwards ir līdzīgs, bet nosūta zvanu uz nosauktu metodi uz GameObject un visiem tā vecākiem.

GAMEOBJECTS ATRAŠANA PĒC NOSAUKUMA VAI TAGA

Vienmēr ir iespējams atrast GameObjects jebkur Ainas hierarhijā, kamēr jums ir kāda informācija, lai tos identificētu. Individuālus objektus var iegūt pēc nosaukuma, izmantojot **GameObject.Find** funkciju:

```
GameObject player;

void Start()
{
    player = GameObject.Find("MainHeroCharacter");
}
```

GAMEOBJECTS ATRAŠANA PĒC NOSAUKUMA VAI TAGA

Objekts vai objektu kolekcija var tikt atrastas pēc to taga, izmantojot **GameObject.FindWithTag** un **GameObject.FindGameObjectsWithTag** metodes.

Piemēram, gatavošanas spēlē ar vienu pavāru un vairākām plīts ierīcēm virtuvē (katru apzīmējot ar "Stove"):

```
GameObject chef;
GameObject[] stoves;

void Start()
{
    chef = GameObject.FindWithTag("Chef");
    stoves = GameObject.FindGameObjectsWithTag("Stove");
}
```

GAMEOBJECTS IZVEIDE UN IZNĪCINĀŠANA

Jūs varat radīt un iznīcināt GameObjects, kamēr jūsu projekts darbojas. Unity, GameObject var tikt izveidots, izmantojot **Instantiate** metodi, kas izveido jaunu kopiju esoša objekta.
Lai iegūtu pilnu aprakstu un piemērus, kā radīt GameObjects, skatiet Prefabu instancēšanu izpildlaikā.
Destroy metode iznīcina objektu pēc kadra atjaunināšanas beigām vai pēc nepieciešamības pēc īsa laika aizkavēšanās:

```
void OnCollisionEnter(Collision otherObj) {
    if (otherObj.gameObject.tag == "Garbage can") {
        Destroy(gameObject, 0.5f);
    }
}
```

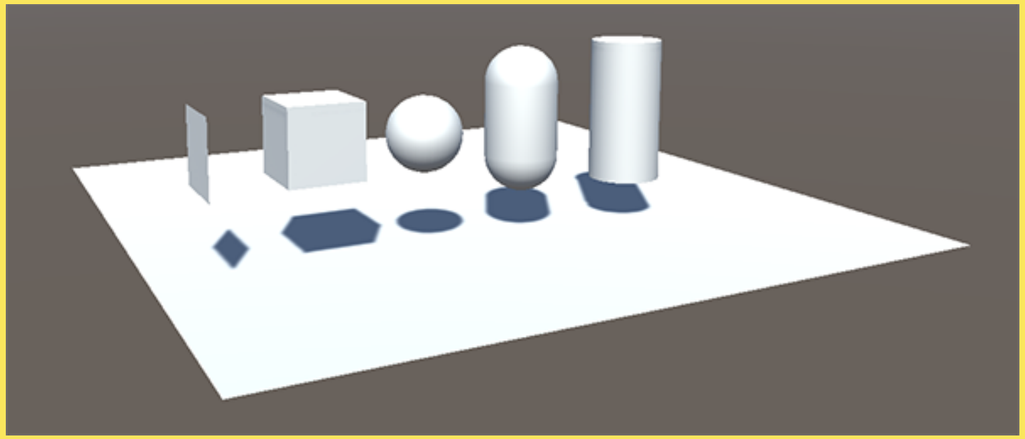
Jāņem vērā, ka Destroy funkcija var iznīcināt atsevišķus komponentus, neietekmējot pašu GameObject. Bieži sastopams kļūdas veids ir rakstīt šādi un pieņemt, ka tas iznīcina GameObject, pie kura skripts ir pievienots:

```
Destroy(this);
```

PRIMITĪVI (PRIMITIVES)

GameObject klase piedāvā skriptu pamatotas alternatīvas opcijām, kas pieejamas Unity GameObject izvēlnē, kas ļauj izveidot primitīvus objektus.

Lai izveidotu Unity iebūvēto pamatobjektu instances, izmanto **GameObject.CreatePrimitive**, kas instancē pamatobjektu no norādītā tipa. Pieejamie pamatobjektu tipi ir *Sphere*, *Capsule*, *Cylinder*, *Cube*, *Plane un Quad*.



Izmantotie informācijas avoti

- https://docs.unity3d.com/Manual/GameObjects.html
- https://docs.unity3d.com/Manual/class-GameObject.html
- https://learn.unity.com/tutorial/activating-gameobjects#5c8a4435edbc2a001f47cdfa
- https://www.youtube.com/watch?v=uCA8Q3nwaz4&ab_channel=CodeMonkey
- https://www.youtube.com/watch?v=2LSXHtl0Mss&ab_channel=ThisisGameDev
- https://www.youtube.com/watch?v=g_Ff1SPhidg&list=PLECF6hIN7gWH3KMJqj3pkOC3mIxBBGRS9&ab_channel=Brackeys
- https://www.youtube.com/watch?v=kH_piLCynto&ab_channel=BoardToBitsGames
- https://starloopstudios.com/what-is-a-unity-gameobject-and-how-do-you-fit-it-into-your-game/
- https://www.javatpoint.com/unity-gameobjects