Luka Funda, bacc. oec.[1]
mr. sc. Ivan Livaja, v. pred.[1]

NOVITETI ALATA UNITY I PRIMJENA VAN INDUSTRIJE VIDEO IGARA

Stručni rad / Professional paper
UDK 004.4

Ovaj rad se nadovezuje na prethodni stručni rad „Osnovni elementi razvojnih alata Unity i Unreal Engine za stvaranje računalnih igara" sa fokusom na Unity. U ovom radu će se govoriti o novitetima koji su se razvili za Unity te će se navesti par primjera praktične primjene programa van industrije video igara.

Ključne riječi: *Unity, game engine, proširena stvarnost, animirani filmovi.*

1. Uvod

Obzirom da svijet, kad je u pitanju tehnologija, napreduje eksponencijalnom brzinom, ljudi imaju sve veće tehnološke zahtjeve, kako općenito, tako i kad su u pitanju video igre. Tvrtke moraju stvarati grafički ljepše i sadržajno bogatije video igre kako bi ostvarili želje svojih kupaca, a da bi udovoljili zahtjevima tvrtki, programi za razvoj igara moraju isto tako pružati veće mogućnosti.

Ono što je zanimljivo i nije moguće primijetiti u početku je to što razvoj tih programa pruža još jednu mogućnost: praktičnu primjenu van industrije video igara. U zadnjih nekoliko godina, sve je veći trend korištenja programa kao što su Unity i Unreal Engine u televizijskim emisijama kao što su vijesti i prognoze vremena, arhitekturi, automobilskoj industriji, filmskoj industriji pa čak i u turizmu.

2. Općenito o Unity-u

Unity je game engine kojeg razvija tvrtka Unity Technologies i koristi se za stvaranje 2D i 3D video igara za različite platforme, kao što su PC, Mac, Linux, Xbox One, Playstation 4 i mnoge druge.

Od verzije Unity 5 pa do danas, na tržištu je moguće pronaći veliki broj naslova raznih game developer[2]-a. Kako je dosta njih neiskusno, Unity se kontinuirano razvijao i dodavao funkcije i inovacije koje su olakšale rad na Unity-u, kao i razne tutoriale [3] koji pokazuju osnove korištenja Unity programa te osnove programiranja. Također, Unity kontinuirano organizira

[1] Veleučilište u Šibeniku
[2] game developer – osoba koja je uključena u sve procese razvoja video ige
[3] tutorial – video koji objašnjava kako koristiti nekakav alat ili razvijati neku vještinu

seminare, tečajeve i „livestream"[4]-ove preko kojih su u kontaktu sa tvrtkama i individualnim korisnicima.

Povećanju popularnosti veliku ulogu igra i sama zajednica, obzirom da dosta iskusnih programera koji koriste Unity objavljuju svoje tečajeve, tutoriale i igre, pa tako osobe koje su zainteresirane ući u industriju video igara imaju na raspolaganju veliku količinu materijala za učenje i vježbanje.

U trenutku pisanja rada, zadnja verzija Unity-a je 2019.1.0f2.

3. Noviteti u alatu Unity

Iako se alat kontinuirano razvija, dodajući poboljšanu kompatibilnost i olakšavanje programiranja kroz razne atribute kao što su „[SerializeField]", „[Range(x,y)]" i „[Header(„naziv")]", mogućnost ulaska u „Debug" mod te mogućnost pristupanja komponentama kroz „GetComponent<>()" i „FindObjectOfType<>()", najveći novitet koji je ostvaren za Unity je Prefab sustav koje je izašao u verziji 2018.3.

Prije verzije 2018.3., programer je mogao stvarati „prefab"[5]-e, što je omogućavalo brzo i lako postavljanje nove scene i izmjene objekata po potrebi: na primjer, promjena brzine rotacije objekta ako postoji skripta za rotaciju, promjena boje i slično. Međutim, korisnik nije mogao stvarati „nested prefabs."[6] Takvi objekti jesu imali predloške drugih objekata, ali su se tretirali kao potpuno novi predlošci.

3.1. Atributi

Prema Unity dokumentaciji, atributi su *„oznake koje se mogu postaviti iznad klase, svojstva ili funkcije unutar skripte kako bi naznačila posebno ponašanje."*[7] Dakle, atributi daju klasama, varijablama i funkcijama dodatne mogućnosti i organizaciju. U skriptama, atributi se stavljaju u uglate zagrade te se stavljaju iznad deklaracije klase, u redu deklaracije varijable ili iznad varijabli, ovisno o tome koji atribut korisnik unosi. Najčešće korišteni atributi u Unity-u su:

- **[SerializeField]** – atribut koji se stavlja uz privatnu deklarirani varijablu. Ovaj atribut tjera varijable koje su privatne da se prikažu u editor-u, a na isti način se mogu stvarati i reference na druge objekte. To omogućava programeru izmjenjivanje vrijednosti tih varijabli radi dorade, kao i olakšanje jer iste ne mora definirati kao javne varijable koje bi drugi korisnici mogli izmjenjivati i tako srušili program. Na isti način se mogu stvarati i reference na objekte.
- **[Header(„naziv")]** – atribut koji postavlja naziv iznad varijabli te se prikazuje u inspector prozoru programa. Glavna funkcija ovog atributa je lakše organiziranje varijabli koje je programer definirao.
- **[Range(x,y)]** – atribut koji definira raspon za neku varijablu. Najviše se koristi za zvuk, kada programer želi postaviti osnovnu vrijednost za jačinu zvuka.

[4] livestream – video koji se prenosi u stvarnom vremenu
[5] prefab – predložak ili nacrt objekta sa svim komponentama i vrijednostima u trenutku stvaranja
[6] Nested prefab – ugniježđeni predložak
[7] https://docs.unity3d.com/Manual/Attributes.html

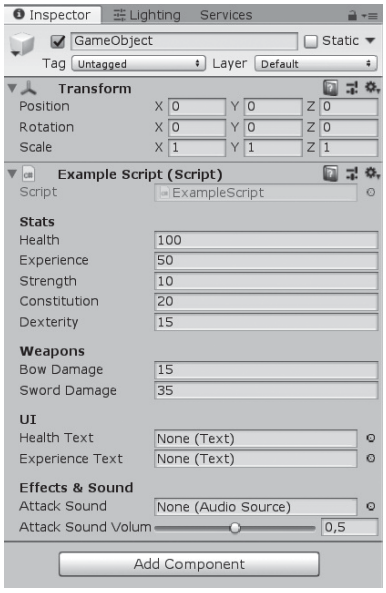
- **[DisallowMultipleComponent]** – atribut koji se stavlja iznad deklaracije klase skripte. Onemogućava postojanje više komponenti te klase u jednom objektu.
- **[Toolbar(„tekst“)]** – ovaj atribut postavlja stvara u editoru pomoćni oblak koji opisuje neku varijablu te je korisna za opisivanje varijabli čiji nazivi nisu razumljivi.

Slika 1: Primjer koda sa korištenim atributima

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

[DisallowMultipleComponent]
0 references | 0 changes | 0 authors, 0 changes
public class ExampleScript : MonoBehaviour
{
    [Header("Stats")]
    [SerializeField] int health = 100;
    [SerializeField] int experience = 50;
    [SerializeField] int strength = 10;
    [SerializeField] int constitution = 20;
    [SerializeField] int dexterity = 15;

    [Header("Weapons")]
    [SerializeField] float bowDamage = 15f;
    [SerializeField] float swordDamage = 35f;

    [Header("UI")]
    [SerializeField] Text healthText;
    [SerializeField] Text experienceText;

    [Header("Effects & Sound")]
    [SerializeField] AudioSource attackSound;
    [SerializeField] [Range(0f, 1f)] float attackSoundVolume = 0.5f;
}
```

Izvor: autorska slika

Slika 2: Rezultat atributa u Unity editor-u



Izvor: autorska slika

3.2. Pristup komponentama objekata

Prije Unity 5, programer je na jednostavan način mogao pristupiti određenoj komponenti objekta. Na primjer, ako programer želi pristupiti RigidBody komponenti, programer je morao samo napisati „rb".Nakon Unity 5, da bi programer pristupio nekoj komponenti, on mora napisati ili **GetComponent<TipKomponente>()** ili **FindObjectOfType<TipKomponente>()**.

Kroz obje naredbe, programer pristupa komponenti koju objekt sadrži, kao i svim svojstvima i funkcijama koje ta komponenta ima. Međutim, glavna je razlika u tome što FindObjectOfType može pristupiti komponentama drugih objekata. Na primjer, programer želi da se u trenutku kada broj života dođe na nulu, pronađe objekt koji sadrži komponentu tipa SceneLoader i zove funkciju koja će korisnika vratiti na scenu sa glavnim izbornikom.

Naredba GetComponent se najčešće koristi u metodi Start, prethodno definirajući varijablu tipa komponente kako bi je mogli cache[8]-irati, iako to nije u nužnosti potrebno raditi. FindObjectOfType se često koristi u funkcijama gdje se zove funkcija koju neki drugi objekt ima.

Slika 3: Primjer korištenja naredbi GetComponent i FindObjectOfType

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.UI;
5
6   [DisallowMultipleComponent]
    0 references | 0 changes | 0 authors, 0 changes
7   public class ExampleScript : MonoBehaviour
8   {
9       //cached reference
10      //reference na komponente na objektu
11      Rigidbody rigidbody;
12      AudioSource audioSource;
13
14      //referenca na komponentu drugog objekta
15      SceneLoader sceneLoader;
16
        0 references | 0 changes | 0 authors, 0 changes
17      private void Start()
18      {
19          // komponente na objektu
20          rigidbody = GetComponent<Rigidbody>();
21          audioSource = GetComponent<AudioSource>();
22          // komponenta na drugom objektu
23          sceneLoader = FindObjectOfType<SceneLoader>();
24      }
```

Izvor: autorska slika

3.3. „Nested Prefabs" sustav

3.3.1. Stari Prefabs sustav

Prije verzije 2018.3., programer je mogao napraviti jednostavan predložak. Taj predložak se potom mogao stavljati u bilo koju scenu te je mogao sa originalnog predloška mijenjati sve „instance"[9] ili primjerke koje su postojale. Također, omogućava brzo postavljanje

[8] cache – privremeni prostor u memoriji koja služi za brzi pristup određenim podacima
[9] Instance – primjerak nekog objekta

objekata u druge scene. Ono što je specifično kod ovog sustava je da promjene napravljene na predlošku neće utjecati na objekte koji su već promijenjeni na bilo kakav način.

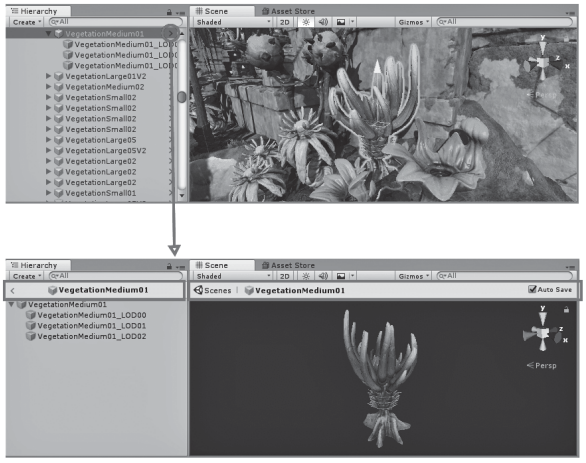
Međutim, kada bi korisnik stavio objekt jednog predloška u drugi objekt te potom iz tog objekta napravio predložak, poveznica s prvim predloškom bi nestala. To nije veliki problem kod razvijanja jednostavnih igara, ali se taj problem postupno povećava sa složenošću i veličinom igre. Ako igra ima veliki broj objekata koji su napravljeni iz velikog broja predložaka, igra postaje puno zahtjevnija po pitanju radne memorije računala.

3.3.2. Novi Prefabs sustav

Novi prefabs sustav omogućava korisniku puno veću funkcionalnost, fleksibilnost i kontrolu nad objektima stvorenih iz predložaka. Također, timovima olakšava i ubrzava razvoj video igara zbog više mogućnosti.

Jedan od noviteta ovog sustava je novi „Prefab Mode". Prema objavi na Unity blogu, *„Prefab Mode omogućava kontroliranu sesiju izmjene Prefab Asset-a jer je moguće raditi sve promjene na Prefab Asset-u, uključujući strukturalne promjene, bez dobivanja prozora koji pitaju za raskid veze sa predloškom."*[10] Dakle, Prefab Mode omogućava korisniku provođenje raznih promjena na nekom objektu stvorenog kao primjerak predloška, a da te promjene ne utječu na predložak.

Slika 4: Prikaz pristupa u Prefab Mode



Izvor: https://blogs.unity3d.com/2018/06/20/introducing-new-prefab-workflows/

Također, novitet su i „Prefab Variants". To su objekti temeljeni na konceptu nasljeđivanja koji se koristi u objektno-orijentiranom programiranju.*„Načelno se čitav postupak svodi na apstrakciju koja polazi od konkretnih podataka do konačnog modela koji se općenito može svesti na klasu s atributima koji opisuju podatke po tipu veličini."*[11]

[10] blogs.unity3d.com/2018/06/20/introducing-new-prefab-workflows/

[11] Urem, F. godina Uvod u objektno orijentirano programiranje s primjerima, Veleučilište u Šibeniku, Šibenik, str. 13.

Dakle, „Prefab Variant" su objekti naslijeđuju atribute izvornog objekta, to jest izvornog predloška, pri čemu je moguće dodavati nove objekte i komponente za dodatnu funkcionalnost. Međutim, postoje određena ograničenja i upozorenja:[12]

- Nije moguće mijenjati strukturu objekata u Variant-i, tj. objekt koji je dijete ne može postati roditelj.
- Nije moguće brisati objekte iz Variant-a koji se nalaze u izvornom predlošku: ako izvorni predložak stabla sadrži nekoliko grana, u Variant-i ih se ne može izbrisati. U tom slučaju, moguće ih je deaktivirati kako bi se dobio isti učinak.
- Ako korisnik uđe u Prefab Mode Variant objekta, sve promjene koje napravi će biti automatski provedene i na izvornom predlošku. Zato je poželjno isključiti opciju „Auto Save".

4. Primjena Unity-a van industrije video igara

Kako se tehnologija ubrzano razvija, tako se i programi namjenjeni za video igru razvijaju. Postali su puno moćniji, pristupačniji i fleksibilniji, to jest, ljudi koji nisu programeri mogu ih koristiti bez problema. Također, u prilog im ide i činjenica da se ti programi mogu koristiti u kombinaciji s drugima, kao što su npr. Gimp, AdobePhotoshop, AutoCAD, Maya i Blender. To su primijetile i tvrtke iz drugih grana djelatnosti, kao što su automobilska industrija, filmska industrija, građevinska industrija te mediji.

U automobilskoj industriji, najviše se koristi za razvoj VR[13] i AR[14] aplikacija, bilo za za razvoj i prikaz novog automobila ili za obuku zaposlenika: *„U Toyoti, koristimo Unity za razvijanje VR i AR alata kako bi poboljšali učinkovitost i kvalitetu dizajna, inženjeringa i obuke. Uistinu je fleksibilna platforma, jer podupire AR/VR uređaje koji nama trebaju."*[15]

Također, vrlo često se koristi u izradi animiranih filmova, u kombinaciji sa drugim programima kao što je Cinema Director: *„Kada kombinirate jake grafičke značajke Unity-a i program za manipuliranje vremenskog toka kao što je naš Cinema Director, stvaranje animacija u Unity-u postaje stvarna mogućnost. ... Unity + Cinema Director kombinacija ima i druge prednosti, moguće je koristiti AI*[16] *i simulaciju fizike za automatiziranje stvari kao što su gužve i uništavanje, smanjujući vrijeme izrade animacije u odnosu na ručno animiranje."*[17] Primjeri takvih filmova i animacija suSonder, ADAM i Mr. Carton.

5. Zaključak

Iako se na prvi trenutak čini kako je Unity program samo za razvoj igara te bi se neiskusnom programeru činio vrlo kompliciran, program je ustvari vrlo jednostavan za korištenje. U tome prilog ide i činjenica da velike automobilske tvrtke kao što su Toyota i Audi za prikaz

[12] docs.unity3d.com/Manual/PrefabVariants.html
[13] VR – „Virtual Reality"; virtualna stvarnost
[14] AR – „Augmented Reality"; izmjenjena stvarnost
[15] unity.com/solutions/automotive-transportation
[16] AI – „Artificial Intelligence"; umjetna inteligencija
[17] cinema-suite.com/5-best-uses-for-unity/

svojih automobila koriste Unity program, kao i za obuku zaposlenika. Također, u kombinaciji sa programom AutoCAD, korisnik može dizajnirati i arhitektonske objekte i interijere, dok uz programe kao što je Cinema Director je moguće stvarati kvalitetne animacije i kratke filmove.

Smatram da će se Unity i drugi programi za razvoj video igara sve više koristiti van industrije za koju su prvotno bili namijenjeni, jer cjelokupna tehnologija postaje jednostavnija za korištenje što doprinosi jednostavnijem i bržem razvoju samih programa.

LITERATURA

1. Urem, F., godina *Uvod u objektno orijentirano programiranje s primjerima*, Veleučilište u Šibeniku, Šibenik
2. www.blogs.unity3d.com/2018/06/20/introducing-new-prefab-workflows/ (pristup: 13.04.2019.)
3. www.cinema-suite.com/5-best-uses-for-unity/ (pristup: 05.05.2019.)
4. www.docs.unity3d.com/Manual/Attributes.html (pristup: 13.04.2019
5. 5.www.docs.unity3d.com/Manual/PrefabVariants.html (pristup: 04.05.2019.)
6. www.unity.com/ (pristup: 13.04.2019.)

Summary

INNOVATIONS IN UNITY AND ITS USE OUTSIDE OF THE GAMING INDUSTRY

This paper continues on the previous paper „Basic elements of Unity and Unreal Engine game engines", with focus on Unity. In this paper, we will talk about some of the innovations developed for Unity and we will mention a few examples of using Unity outside of the gaming industry.

Keywords: *Unity, game engine, expanded reality, animated films.*