Osnove virtualnih okruženja

Laboratorijske vježbe

Vježba 2
Graf scene: Unity3D

FER – ZTE – Igor S. Pandžić
Suradnja na pripremi vježbe:
Nina Kolarec
Mirko Sužnjević
Matko Šilić
Sara Vlahović

1. Uvod

Graf scene je hijerarhijska struktura u koju se sprema virtualna scena na organiziran i strukturiran način. Ona je po svojoj strukturi orijentirano stablo čiji su čvorovi elementi scene. Unity3D je besplatni softver posvećen najviše izradi videoigara, kao što su *Hearthstone, Ori and the Blind Forest* i *Angry Birds 2*, koji sadržava takvu hijerarhijsku strukturu virtualne scene. Osim kompletne virtualne scene, u njemu je moguće definirati i izvore svjetla, položaj kamere, animaciju, efekte, zvuk i sve ostalo što čini virtualnu scenu. Softver je veoma jednostavan i intuitivan te je među najpopularnijim korištenim softverima za razvoj igara.

Cilj vježbe je pomoću sustava Unity kroz ove upute te uz službenu dokumentaciju, dodatne materijale i rješavanjem konkretnih zadataka, dobiti bolji uvid u graf scene i njegov osnovni koncept.

2. Alati potrebni za izvođenje vježbe

Za izvođenje ove vježbe potreban je program Unity3D i Microsoft Visual Studio 2019. Ove upute pisane su za inačicu 2019.3.14f1. Instalacija te inačice obavlja se unutar platforme Unity Hub, na način da se odabere u glavnom izborniku pod opcijom *Installs*. Nakon odabira instalacije te inačice, potrebno je odabrati module Microsoft Visual Studio Community 2019, Windows Build Support te opcionalno dokumentaciju.
Ako koristite macOS ili Linux operacijski sustav, za *build* projekta na vlastito računalo odaberite umjesto Windows Build Support sukladnu podršku za *build* projekta koji odgovara operacijskom sustavu na vašem računalu.

2.1. Izvođenje vježbe na vlastitom računalu

Nakon što preuzmete i instalirate softver prema gornjim uputama, projekt koji ste preuzeli sa stranice predmeta potrebno je raspakirati i dodati ga u Unity Hub te mu pridružiti prikladnu verziju. Projekt tada otvarate unutar programa Unity Hub ili klikom na inačicu sustava Unity koje ste instalirali i naknadnim odabirom. Moguće je da će vam neki predmeti biti ružičaste boje zbog postavki grafičke kartice, no to je potrebno zanemariti kao i upozorenja da neki predmeti su automatski postavljeni na *null* vrijednost.

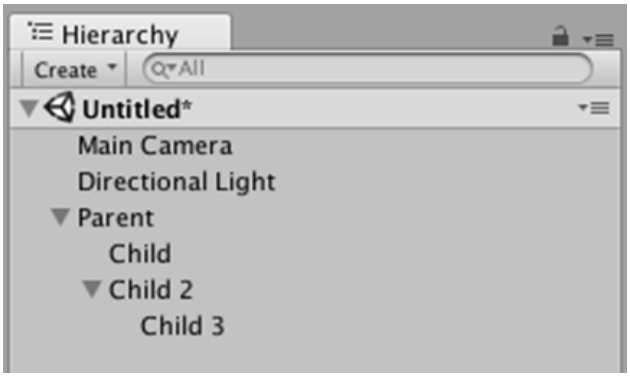
3. Teorijska podloga

3.1. Struktura grafa scene

Graf scene je način organizacije scene koja ima hijerarhijsku strukturu te omogućava pohranu čitave virtualne scene. Njegova struktura predstavlja aciklički usmjereni graf, odnosno stablo. Usmjereni graf znači da ima smjer, a aciklički da postoji samo jedan smjer između dva čvora.
Postoje tri vrste čvorova u stablu: ishodišni (korijenski), unutarnji (račve) i vanjski (listovi). Osim toga, čvorovi mogu biti čvorovi-roditelji i čvorovi-djeca ovisno o nekom drugom čvoru.

Korijenski čvorovi služe za dohvat početne točke scene ili dijela scene te mogu sadržavati neke globalne podatke o sceni, kao što je vrsta koordinatnog sustava. U Unity sustavu je to scena, odnosno *Scene* i može sadržavati više takvih čvorova.

Račve organiziraju scenu u hijerarhiju. Primjer takvih čvorova u pokretačkom sustavu Unity su igrajući objekti (*GameObjects*) s komponentom *Transform* koji predstavljaju transformacijske čvorove te se njihova hijerarhijska struktura može vidjeti u prozoru *Hierarchy*.



Slika 3.1. Prikaz strukture grafa scene u sustavu Unity

Čvorovi listovi su krajnji čvorovi i nemaju djecu. Dok je funkcija čvorova račvi da organiziraju strukturu, funkcija čvorova listova je da sadrže sve elemente scene, a to su uglavnom modeli. Listovi scene se mogu klonirati, no tada umjesto postojanja dvije definicije objekta, zapravo je samo jedna s referencom od dva različita roditeljska čvora.

3.2. Tipovi čvorova

Postoje nekoliko tipova čvorova za listove i račve. Kod listova to su mreža trokuta, kamera i izvor svjetlosti te su u sustavu Unity prikazani kao zasebne komponente *Meshes*, *Camera* i *Lighting*.

Mreža trokuta je struktura koja služi za prikaz geometrije u interaktivnoj 3D grafici. Izvor svjetlosti i kamera se mogu dodati kao zasebni čvorovi u sceni. Oni nam služe za postizanje bolje realizacije scene. Primjerice, ako želimo da kamera prati igrača ona se može postaviti kao dijete transformacijskog čvora.

Kod račvi su sljedeći tipovi čvorova:

- grupirajući – za grupiranje jednog ili više čvorova djece bez dodatnih funkcionalnosti
- transformacijski – grupirajući čvor s mogućnošću geometrijske transformacije (skaliranje, rotacija, translacija), u sustavu Unity to je predstavljeno komponentom *Transform*
- izbornik – grupirajući čvor u kojemu je uvijek aktivno samo jedno dijete tog čvora u bilo kojem trenutku, a bira se numeričkim parametrom
- razina detalja – posebna vrsta izbornika u kojem se izbor djeteta čvora izvodi automatski
- pano – služi za automatsku orijentaciju predmeta prema kameri, primjer toga u sustavu Unity je *Canvas*

3.3. Prolaz kroz graf scene

Prolaz kroz graf scene se uvijek započinje u korijenskog čvora scene i rekurzivno prolazi po hijerarhiji scene. Primjeri operacija koje obavljaju takav način prolazak kroz graf scene su:

- iscrtavanje scene - *Rendering* u sustavu Unity
- test presjeka/sudara - *Collision* u sustavu Unity
- traženje pojedinog elementa

Pseudokod za rekurzivni postupak izgleda ovako:

```
processNode(node, state) {
	localState = updateNode(node, state); // računanje novog stanja
	switch(node.type) {
		case <XYZ>: // posebna obrada za vrstu čvora <XYZ>, npr. crtanje
		...
	}
	for(i = 0; i < node.numberOfChildren; i++) {
		processNode(node.child[i], localState) // rekurzivni poziv funkcije
	}
}

updateNode(node, state) {
	StateType localState;
	localState.gt = state.gt • node.t; // globalna transformacija
	... // računanje ostalih parametara i slično
	return localState
}
```

Rekurzivna funkcija kao početne parametre prima ishodišni čvor i inicijalno stanje te izračuna novo stanje pomoću funkcije *updateNode()* u kojoj je najvažnija matrica transformacije. Globalna matrica transformacije, koja je dio stanja, množi se s lokalnom i na taj način se lokalne koordinate čvora uvijek mogu pretvoriti u globalne jer se zadržava globalna matrica transformacije. Nakon računanja i spremanja novog stanja u varijablu *localState*, vrši se obrada trenutnog čvora.

U pokretačkom sustavu Unity poželjno je da je dubina hijerarhije što manja i da, umjesto jedne velike hijerarhije, bude više manjih. Razlog tomu je upravo prolazak kroz čvorove. Naime, ako se samo jedan transformacijski čvor promijeni, Unity mora proći kroz sve čvorove djece da obavijesti o promjeni. Stoga, ako imamo sto manjih hijerarhija i promijenimo samo jedan objekt u nekoj hijerarhiji, prolazak će biti samo u toj hijerarhiji. U slučaju jedne velike hijerarhije, prolazak će biti kroz sve čvorove kako bi ih obavijestio o toj promjeni. Osim toga, Unity će pomoću više dretvi obavještavati o promjenama počevši od korijenskog čvora. Što više hijerarhija postoji, to će biti više dretvi i cijeli proces prolaska kroz hijerarhiju će se ubrzati.

4. Opis zadatka

Vaš zadatak u ovoj vježbi je implementirati igru u danom projektu koja simulira navedene koncepte grafa scene. Igra se sastoji od lopte koja skuplja predmete na sebe kako bi povećala svoju masu i veličinu te se pritom njena hijerarhijska struktura mijenja, odnosno predmeti koji se lijepe na nju postaju dio njene hijerarhije. Projekt je inspiriran postojećom igrom *Katamari Damacy*.

ZADATAK
Potrebno je implementirati lijepljenje predmeta na loptu (predmeti postaju djeca lopte)

- Lijepljenje predmeta manje mase od lopte, pri čemu se masa lopte povećava
- Implementirati hijerarhiju kako izgleda u završnoj verziji igre
- Prikazati ukupnu masu lopte na ekran (primjer na slici 4.1)
- Prikazati na ekranu hijerarhiju koristeći princip prolaska kroz graf scene (primjer na slici 4.1)

Slika 4.1. Prikaz panoa s ispisom mase i dobivene hijerarhije – na loptu su se prilijepili banana i kikiriki – djeca lopte Katamari; na kikiriki se prilijepila još jedna banana (dijete kikirikija), a na nju jagoda (dijete banane).

DODATNI ZADATAK (NIJE OBAVEZNO)

- Micanje predmeta iz hijerarhije ako se sudare s predmetom veće mase, pri čemu je potrebno osigurati da njihova masa bude ista kao i prije nego što postanu dio lopte, a masu lopte je potrebno smanjiti za tu vrijednost.

Napomena: Dodatni zadatak je također potrebno uključiti u izvještaj.

5. Upute za rad

Na stranicama predmeta je prazan projekt u kojemu je potrebno implementirati funkcije u skriptama *PlayerCollision* i *PlayerInformation*. Projekt treba preuzeti sa završnom verzijom igre koja prikazuje konačan rezultat implementacije projekta. Dodatni komentari za obavljanje zadataka nalaze se unutar skripti.

ZADATAK

- Preuzeti igru sa stranica predmeta i pokrenuti je na vlastitom računalu (pokrenuti datoteku *Katamari.exe* unutar direktorija *Katamari*)
- Preuzeti projekt u kojem je potrebno implementirati i dodati ga u Unity Hub
- Nakon otvaranja projekta u Unity editoru, otvoriti skripte na način da prvo otvorite *solution* projekta u MS Visual Studio 2019 (u Unity editoru *Menu → Assets → Open C# Project* i odabrati u projektu *Katamari.sln*)
- Za testiranje igre u Unity editoru je potrebno kliknuti na gumb ▶ (ponovnim klikom na isti gumb se zaustavlja igra)
- Implementacija lijepljenja predmeta na kotrljajuću loptu na način da nakon kolizije s tim predmetom lopta postane roditelj tog predmeta, ako je predmet manje mase i označen kao „ljepljiv“, odnosno ima oznaku *Sticky* (svi predmeti koji imaju oznaku *Sticky* su pod objektom *StickyItems*)
- Povećanje mase lopte za masu predmeta koji je postao dio njene hijerarhije

- Implementirati hijerarhiju tako da i predmeti koji su djeca lopte mogu postati roditelji novim predmetima ako su oni prilikom kotrljanja uspostavili kontakt (ako je novi predmet prilikom kolizije uspostavio direktan kontakt samo s loptom, novi predmet postaje dijete lopte, ali ako je uspostavio kontakt s nekim od predmeta zalijepljenih na loptu, postaje dijete tog predmeta, a ne lopte) – ovime se dobiva dublja hijerarhija, poput one na slici 4.1. gdje je lopta roditelj kikirikiju, kikiriki banani, a banana jagodi. Pri tome koristiti funkciju *GetContact* od komponente *Collider* za detekciju koji se predmet u hijerarhiji lopte sudario.
- Implementirati pano (*Canvas*) koje će tijekom igre ispisivati ukupnu masu lopte na ekran (skripta *PlayerInformation*)
- Postojeći pano nadogradite opcijom ispisivanja trenutne hijerarhije na ekranu (koristiti funkciju *GetComponentsInChildren* od *GameObject*).

DODATNI ZADATAK (NIJE OBAVEZNO)

- Implementirati micanje predmeta iz hijerarhije koristeći skriptu *PlayerCollision* – predmeti prilijepljeni na loptu će se odlijepiti s lopte te maknuti iz hijerarhije ako se prilikom kotrljanja sudare s predmetom veće mase, pri čemu je potrebno osigurati da njihova masa bude ista kao i prije nego što postanu dio lopte, a ukupnu masu lopte je potrebno smanjiti za tu vrijednost.

Napomena: Dodatni zadatak je također potrebno uključiti u izvještaj.

6. Pomoćni materijali za izradu vježbe

Materijali potrebni za izradu vježbe nalaze se na web stranicama predmeta. Detaljna specifikacija funkcija i dodatnih funkcionalnosti u sustavu Unity nalaze se na službenim stranicama dokumentacije.

7. Predaja rezultata vježbe

Rezultati vježbe se predaju korištenjem aplikacije *Moodle*, dostupne preko web stranica predmeta, u obliku PDF izvještaja koji treba sadržavati:

- opis na koji način ste implementirali zadatke i funkcije koje ste koristili
- kopiran tekst (**ne screenshot**) kompletnih skripti koje ste izmijenili ili napisali
- slike sučelja i postavki komponenti, gdje je potrebno
- *screenshotovi* koji pokazuju implementaciju svakog dijela zadatka (i samog *Play* ekrana i rezultirajuće hijerarhije koja pritom nastaje unutar sučelja Unityja)
- *screenshot* ostvarenog prikaza hijerarhije

Napomena: Rezultati se šalju isključivo preko gore navedene aplikacije. U slučaju problema, javiti se email-om na adresu vo@fer.hr. Sačuvajte kopiju poslanih rezultata.