

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike u Puli

Beyond Reach

Dokumentacija

Kolegij: Izrada Informatičkih Projekata

Mentor: doc. dr. sc. Nikola Tanković

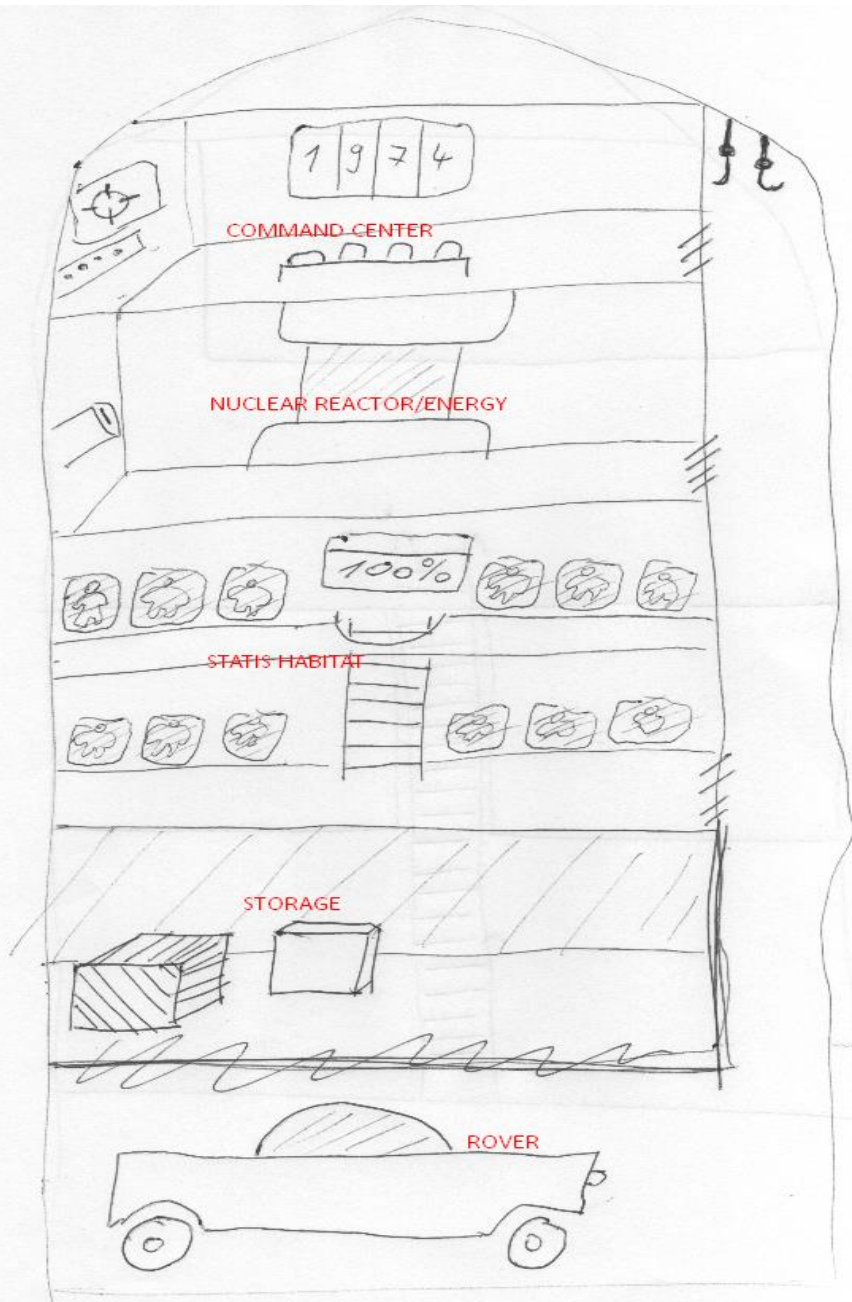
Student: Josip Marić

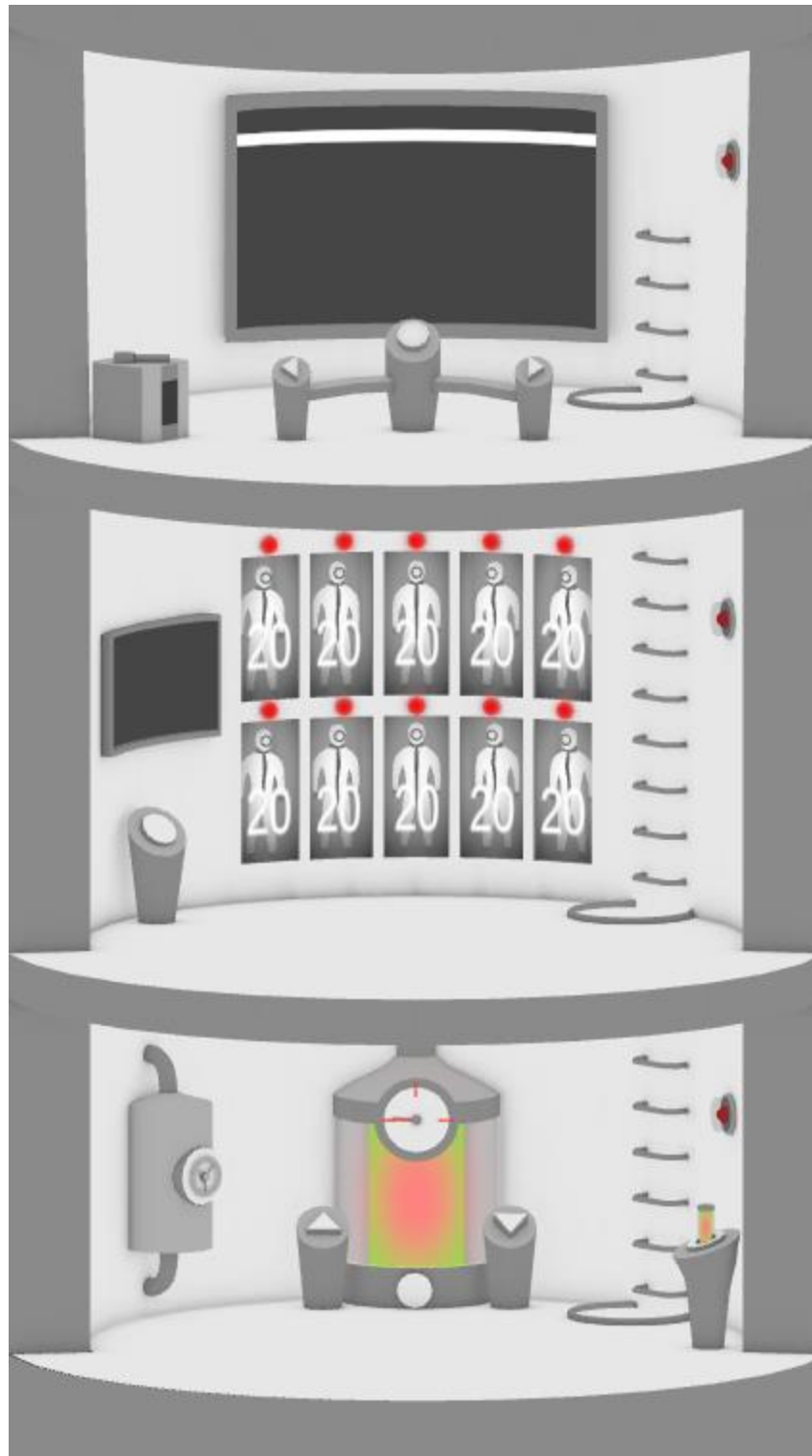
1. Sažetak

Sljedeći tekst objašnjava koji su se programi koristili za izradu igre, opisuje gameplay igre te klasni dijagram. Nadalje, opisuju se glavne funkcionalnosti i kako su neke od tih funkcionalnosti implementirane.

2. Uvod

Igra je izrađena u Unity razvojnom okruženju, korišten je C# programski jezik i ECS arhitektonski uzorak. Igra je izrađena za mobilni uređaj kao prvobitnu platformu. Cilj igre je održavanje rakete do njene posljednje destinacije. Gameplay igre je nekih 20 minuta i sastoji se od zagonetki koje se rješavaju preko point-click mehanike. Igra je izrađena tako da se svaki pojedini događaj zapisuje te prati se djelotvornost subjekta te se na temelju toga konstantno kroz igru izmjenjuje težina. Također izrađen je i uređivač preko kojeg je moguće "pisati" priču koja će se prikazati igraču.





3. Opis Igre

Započinjete dolaskom na raketnu platformu. Tamo nema nikoga, svi su već u zastoju unutar rakete. Vaša misija je usmjeriti raketu na Mars. Započinjete pokretanjem rakete. Sljedeća stanica je komandno središte. Kad stignete tamo, morate probiti kod za lansiranje rakete. Polijećeš. Odjednom se oglasi alarm. U međuvremenu je potrebno podesiti pritisak na motoru rakete kako bi se izbjeglo topljenje. Stvari se smiruju, uskoro se uključuje novi alarm. Potrebno se spojiti sa orbitnom stanicom kako bi se prikopčala sekcija sa reaktorom. Potrebno je odraditi docking proceduru te pri neuspjehu dolazi do sudara sa stanicom pri čemu će raketa pretrpjeti blago oštećenje te bit će potrebno ponovno odraditi proceduru do uspjeha.

Pri uspješnom obavljanju operacije, sljedeće uz postavljanje kursa prekida se orbita i kreće se prema Mjesecu. Na putu do tamo udarit će vas oluja Sunca koja će vam oštetiti većinu sustava. Jedan od njih je komora za stazu, koji više ne može izdržati sve ljude unutra pa ćete morati žrtvovati jednog da biste spasili ostale. Ploča u sredini sobe govori koliko je vjerojatno da će kolonija preživjeti kad stignu na Mars. Na putu do Mjeseca održavate sustave, tlak u reaktoru i prilagođavate kurs. Kad se nađete u Mjesečevoj orbiti, kontaktirate lokalnu stanicu kako bi obnovili gorivo. Nakon što je sve to učinjeno i svi drugi sustavi su u funkciji, prekinete orbitu i krenete prema Marsu. Na putu do tamo popravljate sustave, održavate stabilnost tlaka u reaktoru i prilagođavate kurs kad iznenada naletite na asteroide. Vaš je posao unositi koordinate kako bi izbjegli asteroide. Za svaki sudar koji raketa pretrpi oštetit će se reaktor. Nakon prolaska kroz asteroidno polje uđete u orbitu Marsa, započinjete postupak kolonizacije (rješavanjem koda) gdje iz reaktora uzimate nuklearno gorivo i stavljate ga u novi otvor u komori za zastoj. Nakon toga pritisnete veliko crveno dugme, a komora za odlaganje i spremište odlijepe se i krenete prema Marsu. Umireš. Završni krediti pričaju vam malu priču ovisno o tome koji su kolonisti preživjeli putovanje, pa čak i ako je netko preživio.

Napomena: Trenutna verzija igre ne slijedi nužno događaje u opisu igre ali koristi sve navedene elemente, mehanike i slično.

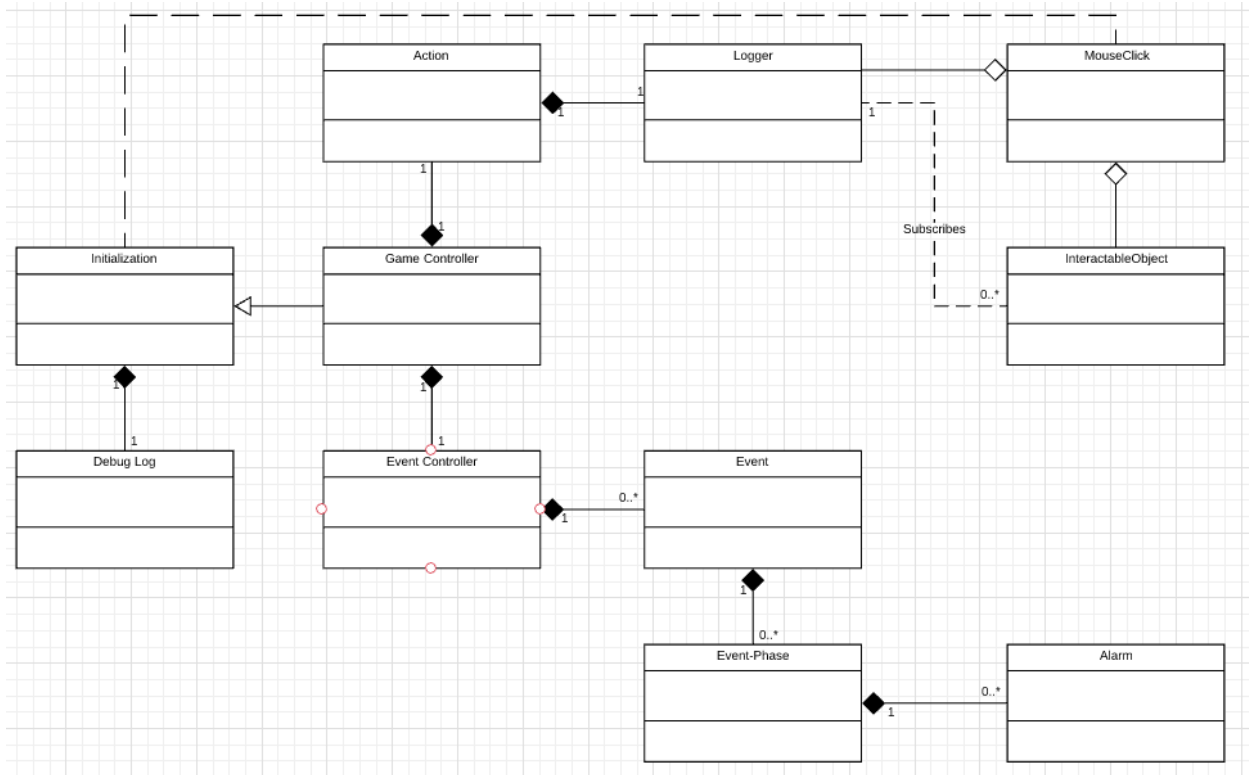
4. Klasni dijagram

Započinje izradom Initialization i Game Controller-a, roditeljske klase, odnosno klase djeteta. Dakle, sve važno, poput referenci objekata, učitavanja, spremanja i pauziranja igre prolazi kroz ovaj kontroler. U inicijalizaciji se instancira sve sekundarne klase (mehanike igre), kao što su zapisnik o otklanjanju pogrešaka (različit od jedinstvenog zapisnika otklanjanja pogrešaka), radnja i kontroler događaja. Zanimljivost vezana uz Action i Event Controller je da oboje stvaraju instance dodatnih klasa te im se može pristupiti iz Game Controllera pomoću spomenutih klasa.

Objekt Action poziva se nakon što upišemo Interactable Object u objekt Logger (Logger je deklariran i instanciran u radnji). Interaktivni objekt pretplaćen je na događaj Click or Tap koji je deklariran u klasi MouseClick koja također sadrži referencu na objekt Logger koji mu je dala klasa Initialization. U biti, ako se klikne na interaktivni objekt u igri, taj se objekt pretplati na zapisnik koji zatim pozove metodu u Action objektu i izvrši određenu operaciju (poput premještanja znaka na taj objekt itd.). Razlog zašto je ovo strukturirano na ovaj način je omogućiti igraču da klikne na više međusobno prihvatljivih objekata u igri kako bi se mogao napraviti popis aktivnosti koje će se izvoditi jedna za drugim. Sada kako bi se dopustilo igraču da poništi akciju (jer je njegovo prisustvo potrebno na drugim mjestima u igri), implementiran je događaj dvostrukog klika ili dvostrukog dodira koji će izbrisati (odjaviti) popis aktivnosti iz Loggera i pretplatiti se na novu radnju.

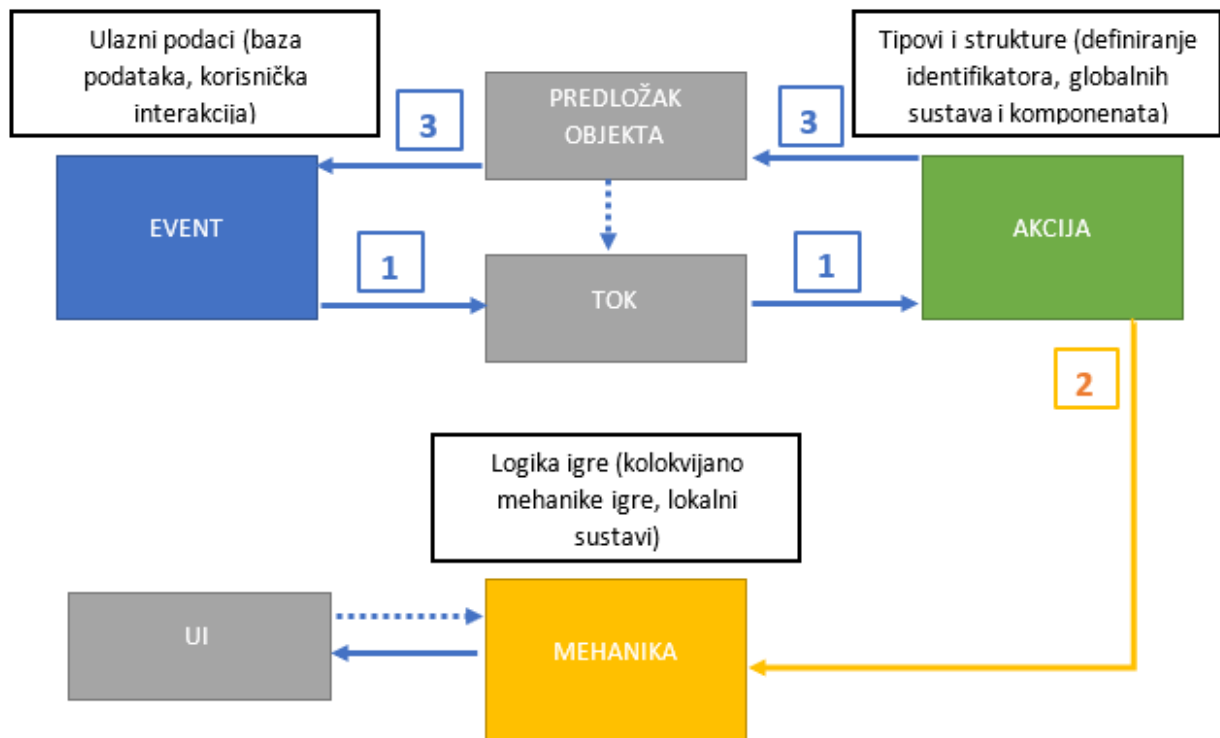
S druge strane, imamo kontroler događaja koji instancira objekt događaja koji zatim čini isto za fazu događaja, a isto za alarm. Ideja je da se pomoću ugrađenog uređivača napravi događaj koji će sadržavati više faza, a svaka faza više alarma. Što znači da ako postoji događaj, možemo odrediti faze koje se aktiviraju pod određenim uvjetima, na primjer, ako igrač odluči otvoriti vatru, neprijatelj bi također mogao ući u ovu fazu događaja. Sada ova faza može sadržavati alarme kao što su oštećenje trupa, onemogućeno održavanje života itd. Sve to može zahtijevati od igrača da reagira.

Posljednja stvar je što će sve ove aktivnosti generirati neku vrstu povratnih informacija, te povratne informacije će se obraditi objektom Debug Log, tako da se mogu koristiti za ispravljanje pogrešaka.



5. Implementacija igre

Po koracima prikazanih na slici gdje se prikazuje put izvođenja pri interakciji korisnika s interaktivnim objektima, ovisno o identifikatorima entiteta (objekta) izvršava se neka od akcija. Akcije se pune u stream modul koji se sastoji od jednostruke vezane liste gdje se akcije izvršavaju jedna nakon druge. U drugom koraku, koji je opcionalan, moguće je utjecati na izvršavanje mehanike preko identifikatora podobjekt što bi se koristilo za interakciju korisnika s vanjskim sustavima (poput korisničkog sučelja). Izvršenjem akcije pokreće se treći korak koji kod postavljenog delegata (ako se izvršila definirana akcija) signalizira Event modulu učitavanje novog cilja (u sklopu sustava misija) i signalizira stream modulu nastavak rada ako ima akcija u njegovom redu čekanja.



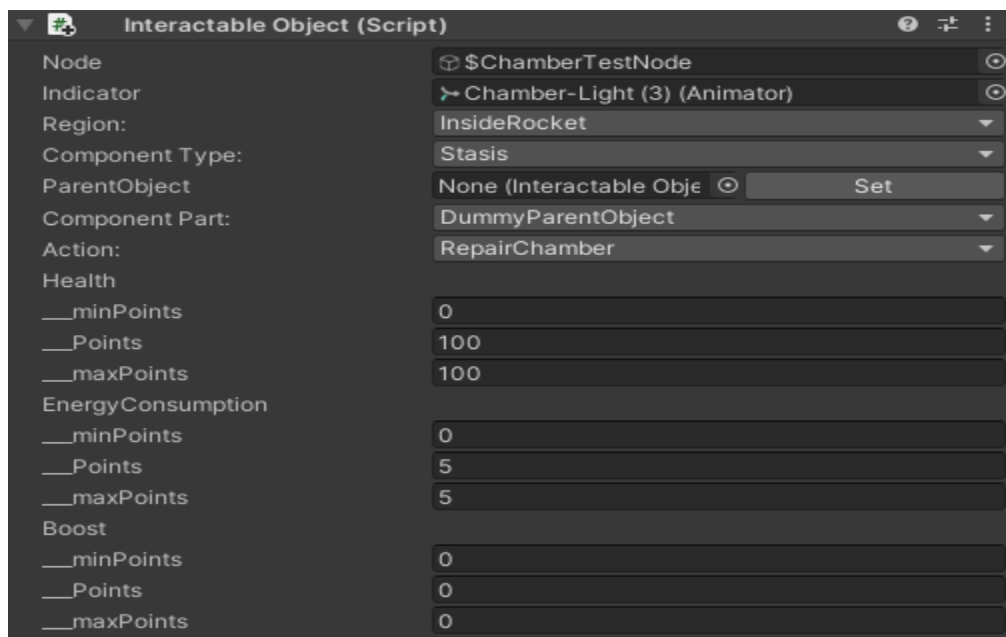
6. Kreiranje entiteta

Nakon definiranja svih entiteta preostaje njihovo dodavanje u scenu igre. Kako se koristi Unity, njihovo postavljanje se olakšava tako da svi entiteti nasljeđuju od *MonoBehavior*.

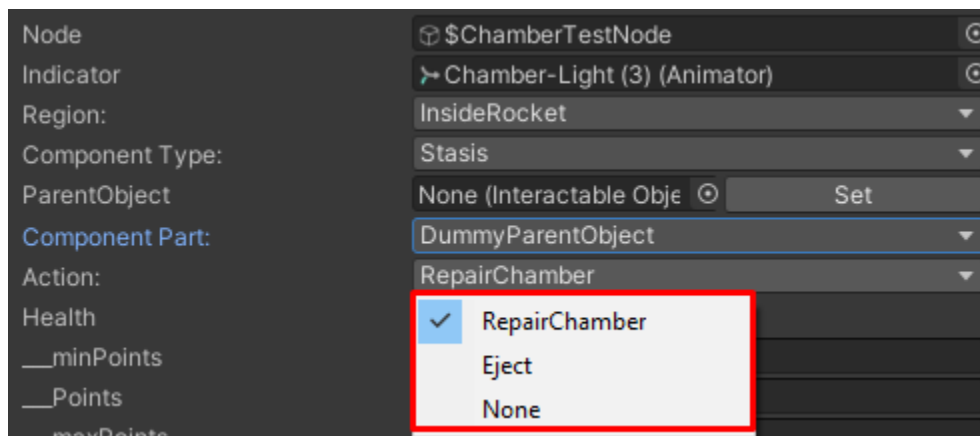
Postoje tri različita tipa entiteta:

- Interactive (hrv. interaktivan)
 - Entitet nad kojim igrač vrši direktne akcije. Preporučljivo je da se u ovom slučaju koriste *akcije* apstraktne klase *interactable*.
- Default (hrv. standardni)
 - Entitet neovisan o igraču, ali za njegovo korištenje treba dodati mehaniku odnosno lokalni pristup sustavima.
- Dependent (hrv. ovisni)
 - Specijalni tip entiteta koji se ne dodaje u sceni, nego je vezan za određenu *akciju* (početak akcije) i nestaje pri izvršenju te *akcije*.

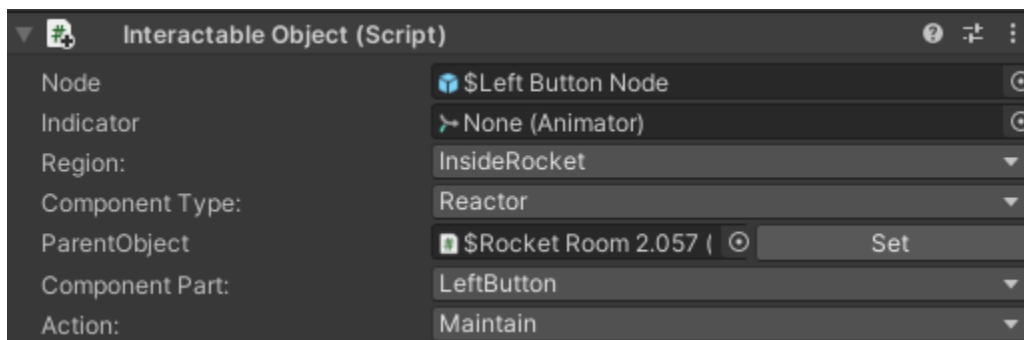
Njihovo postavljanje je osnovno dodavanje skripte u Unity igračem objektu (eng. game object) i postavljanja parametara. *Node* i *indicator* služe pri komuniciranju s ostalim ugrađenim sustavima razvojne platforme, a ostalim se koristi ECS.



Kako su za „stasis“ (hrv. komora za zastoje) definirane samo *akcije* „repair chamber“ (hrv. popravi komoru), „eject“ (hrv. izbacij) i „none“ (hrv. ništa), tako su nam one na raspolaganju. Isto vrijedi i za *podobjekte* (koji su u sučelju nazvani component part).




Ako bi se nadalje dodao entitet s *podobjektom* koji nije roditelj ili lažni roditelj, ali njegov roditelj (entitet) bi bio tipa *interactable* tada bi unos parametara. Dodatna prednost više identifikatora je u tome što ako se prvo doda entitet roditelj, moguće je klikom na gumb „set“ automatski postaviti referencu na entitet roditelja tog tipa *objekta*.



7. Implementacija uređivača

Uređivač je izgrađen u Vue 2.0 okruženju, TypeScript programskom jeziku. Aplikacija je isključivo front-end te koristi se Sqlite baza podataka za pohranu krajnjeg rezultata kojeg se preuzme i ubaci u igru. Trenutno potrebno je ponovno izgraditi igru nakon mjenja baze.

Home



Spremi Bazu

Cyber Database Editor

0 +

0 Refuel +

Add Phase

Edit

Action

Dialogue ▾

Repeat

1

Save

Edit Remove

Action

Avoid ▾

Repeat

1

Save

Edit Remove

Action

Course ▾

Repeat

1

Save