Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/17BE16Ivkovic

PLATFORMA ZA SPECIFIKACIJU I INTERPRETACIJU INTERAKTIVNE FIKCIJE

PLATFORM FOR INTERACTIVE FICTION SPECIFICATION AND IMPLEMENTATION

Aleksa Ivković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj rad predstavlja implementaciju jezika specifičnog za domen za potrebe kreiranja igri iz žanra interaktivne fikcije. Ovaj projekat je realizovan pomoću biblioteke textX. U okviru rada istražene su karakteristike ovog žanra, kao i neka slična rešenja iz domena.*

Ključne reči: *Jezici specifični za domen, interpreter, interaktivna fikcija*

Abstract – *This paper presentes implementation of a domain specific language for creating interactive fiction games. This project was done using textX library. Also this paper explores characteristics of this genre, as well as some similar solutions in the domain.*

Keywords: *Domain specific languages, interpreter, interactive fiction*

1. UVOD

Personalni računari nisu u masovnoj upotrebi ni 50 godina, ali napredak koji su oni dostigli za taj period je neverovatan. Takav napredak nije zabeležen do sada u istoriji čovečanstva. 70-ih godina prošlog veka pojavili su se personalni računari, a na njihovu revolucionarnu pojavu je dodatno uticao momenat njihove svrhe i upotrebe - ljudi su dobili priliku da ih koriste u udobnosti svog doma. U tom periodu računari su imali svega nekoliko kilobajta memorije i softveri su morali da budu optimizovani kako bi mogli da izvuku njihov maksimum. Samim tim igre sa grafičkim interfejsom nisu ni bile moguće. Shodno tome, na tržištu su se pojavile igre koje su pripadale žanru interaktivne fikcije. One su bile dosta kompleksne i pružale su veliki broj sati zabave igračima [1]. U okviru ovog rada će biti istražene same igre i njihov istorijat, ali će veći fokus biti na samom načinu njihove implementacije.

2. INTERAKTIVNA FIKCIJA

Interaktivna fikcija (u daljem tekstu IF) u svom najširem smislu može predstavljati bilo koji vid pripovedanja u kome čitalac ili slušalac aktivno učestvuje. U okviru ovog rada će se posmatrati jedan određeni tip IF, koji je ujedno i najpopularniji, a to jeste u vidu igri za računare. Neki vid definicije ovih igara je dat od same zajednice koja je pomogla da ovaj vid zabave ne izumre: „IF predstavlja jedinstveni vid kompjuterskog pripovedanja koji stavlja igrača u ulogu lika u simuliranom svetu, koji karakteriše njegovo oslanjanje na tekst kao glavni vid opisa, a koristi fleksibilni jezik, sličan ljudskom, za izdavanje komandi, koje obrađuje parser“ [2].

2.1. Počeci

Začetak ovog žanra predstavlja igra pod nazivom Hunt the Wumpus. Ova igra je nastala 1972. godine od strane Gregory Yob – a. Ono što nju razlikuje od njenih naslednika je to što nije sadržala parser koji je mogao da tumači kompleksne rečenice, već je igrač imao ograničen skup akcija koje je mogao da izvrši. Ove akcije su se sadržale od dve reči, glagola i objekta.

Sredinom 1970 – ih, nastala je igra Adventure, prva ovog žanra koja se odlikuje svim karakteristikama modernog IF – a. Njen kreator je Will Crowther sa Stanford – a, inspirisan, tada novom društvenom igrom, Dungeons and Dragons. Advenutre je igra napisana u Fortran – u i bila je kreirana za PDP-10. Veliki korak je napravljen ovom igrom, jer je Will kreirao parser koji je mogao da obrađuje naredbe od dve reci, sa mnogo većim vokabularom nego u igri Hunt the Wumpus.

2.2. Komercijalna era

1978. godine, Scott Adams, pod uticajem igre Adventure, odlučuje da je modifikuje, kako bi mogla da se izvršava na kućnim računarima, koji su imali znatno manje memorije. Tako je nastala igra poznata pod nazivom Adevntureland. Kako bi postigao ovakav cilj Scott je kreirao svoj jezik i interpreter, kako bi sama igra bila nezavisna od platforme na kojoj se izvršava. Sa ovakvim rešenjem, Scott je postigao veliki uspeh.

Paralelno sa razvitkom Adventureland – a, nastaje i igra pod nazivom Zork. Naime, u trenutku kada je Advenutre pušten u ARPANET, grupa studenata sa MIT – a je pronašla igru i odlučili su da naprave bolju igru od nje. Od 1975. do 1979. su razvijali ovu igru. Igra Zork ne predstavlja komercijalni uspeh, već tehnološki, njen parser je bio u mogućnosti da prepozna složenije fraze, i imao je vokabular od oko 900 reči. Svi moderni parseri koriste sličnu paradigmu kao i ovaj parser.

1979. godine su asistent sa MIT – a i neki od začetnika Zork igre odlučili da osnuju svoju kompaniju Infocom, primarni fokus kompanije je bio prilagoditi igru Zork kućnim računarima. Zaposleni u kompaniji su shvatili da mogu da iskoriste novu moć personalnih kompjutera, tako što će samo izvršavati igru na računarima. Ovo su postigli pomoću kompjuterskog čipa Z-machine koji su kreirali.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Igor Dejanović, vanr. prof.

Na ovom čipu bi igra mogla da se izvršava, ali da bi zapravo mogla da se izvršava na računaru, svaki uređaj je morao da ima ZIP (Z-machine Interpreter Program). Ovaj interpreter bi omogućio da se igra izvršava isto kao i na Z-machine. Ali kako bi ovo sve bilo moguće, bilo je neophodno kreirati jezik koji je dovoljno kompaktan kako bi kompleksna igra mogla da stane, a i da se izvršava na uređajima tog vremena. Tako je nastao jezik Zork Implemented Language (ZIL). Igra Zork je donela veliki uspeh kompaniji, tako da su nastala dva naslednika Zork II i Zork III, kao i oko 30 drugih naslova.

2.3. Kraj ere i trenutno stanje

Nakon uspešne decenije, početkom ’90 – ih godina prošlog veka, IF žanru počinje da opada popularnost. Naime, razvitkom računara, i većom snagom i kapacitetom koji su oni doneli, same računarske igre su mogle biti zahtevnije. U tom periodu se pojavljuje novi tip avanturističkih igara, koji se u potpunosti oslanja na vizualni prikaz i kliktanje mišem. Pojavile su se dve nove kompanije koje su dominirale tržištem avanturističkih igara baziranih na igri pomoću miša, a to su bile Sierra i Lucasarts. Ljudi su bili očarani novim grafičkim dostignućima i samim tim IF igre im nisu bile više primamljive.

Tokom 1990 – ih i 2000 – ih bilo je više pokušaja da se oživi IF žanr u komercijalnom smislu, ali je to bilo bez nekih većih uspeha. Do danas žanr održava njegova mala grupa vernih obožavaoca, koji razvijaju svoje naslove i većinski ih besplatno distribuiraju na internetu. Postoji nekolicina foruma i blogova koji omogućavaju istomišljenicima da razmenjuju ideje i pomažu jedni drugima da sačuvaju ovaj žanr od zaborava.

3. SLIČNA REŠENJA

3.1 Alan

Alan predstavlja jezik specifičan za domen baziran na karakteristikama objektno orjentisanog programiranja. Prva verzija ovog jezika je nastala 1985. godine kada je ovaj žanr bio u svom vrhuncu. Tokom godina je unapređen i trenutno je aktuelna treća iteracija ovog softvera. Ovaj jezik je namenjen autorima IF igara, tako da je njegova sintaksa približena više ne tehničkim licima. Glavna ideja tokom razvoja ovog jezika jeste jednostavnost, ali time nije uskraćena implementacija kompleksnijih ideja. Sintaksa ovog jezika se može posmatrati više kao skup opisa koji nalikuju rečenicama [3].

3.2 Adventuron

Adventuron predstavlja softversko rešenje koje svojim korisnicima pruža online editor za tekst, i kompajler koji može da izvršava igre u pretraživaču, na računari, mobilnim telefonima… Za razliku od prethodnog jezika, Adventuron je okrenutiji programerima svojom sintaksom. Ona najviše podseća na hibrid JSON formata i JavaScript jezika [4].

4. JEZICI SPECIFIČNI ZA DOMEN

Jezik je sistem znakova, simbola, gestova i pravila koji se koriste u komunikaciji. U kontekstu računara, jezik predstavlja način komunikacije između korisnika i računara, ili između dva računara. Programski jezici se sastoje od skupova stringova koji se kompajliraju u komande koje računar može da izvrši. Jezike možemo podeliti na jezike opšte namene (engl. *general purpose languages*) i jezike specifične za domen (engl. *domain specific languages*). Neki od poznatih jezika opšte namene su: C, C#, Java, Python, JavaScript. Jezici opšte namene su definisani na način koji im omogućava da se prilagode velikom broju domena, samim tim i način na koji se može doći do rešenja je velik. Ova karakteristika nosi i lošu posledicu sa sobom, a to jeste, povećana šansa za greškama[5].

Da bismo mogli da razumemo namenu jezika specifičnih za domen, neophodno je da razumemo šta domen predstavlja. Domen predstavlja sfere delovanja, interesa ili funkcije. Jezici specifični za domen su prilagođeni jednom domenu i oni mogu da se koriste vokabularom i notacijama tog domena. Iz razloga što su ovi jezici usko vezani za njihove domene, drugi naziv im je i mali jezici.

Odabirom nekog jezika specifičnog za domen za realizaciju softvera, čini se prvi korak ka efektivnijem rešenju. Naime, studije su pokazale da se korišćenjem ovakvih jezika produktivnost može uvećati i do 10 puta. Pored toga, ovi jezici su koncizniji, a takođe mogu da se maksimalno prilagode potrebama korisnika. Kada se pristupa rešavanju problema, neophodno je kreirati konceptualni model za koncepte iz domena problema. Najveći problem ovog procesa jeste ručno mapiranje koncepata, na koncepte domena rešenja, pri korišćenju jezika opšte namene [5].

5. TEXTX

TextX predstavlja alat koji je izgrađen na Arpeggio – vom parseru, ali je njegova gramatika jednostavnija, konciznija i ekspresivnija [6][7]. Rad textX – a je baziran na sledećem toku, prvo se parsira sama gramatika jezika. TextX dinamički kreira meta – model i Arppegio parser. Parser će dalje parsirati sam model i kreirirati graf Python objekata u odnosu na meta – model. Model se dalje može koristiti u svrhe interpretacije ili za generisanje fajlova.

6. OPIS REŠENJA

6.1. Jezik

Prvo će biti opisan sam jezik, njegove osobine i sama struktura. Pre toga, potrebno je napomenuti da je sama sintaksa bila inspirisana Python jezikom i sintaksom online alata Adventuron, koji je bio opisan u trećem poglavlju. Zamisao ovog rada jeste bila stvoriti jezik koji pruža korisniku veliku fleksibilnost pri pisanju igre, i time mu omogućiti da pomoću nekih elemenata koje sam definiše, bude u mogućnosti da postigne željene rezultate. Glavne komponente koje omogućavaju funkcionalnost ovog softverskog rešenja jesu: lokacije, veze, objekti, stanja, akcije i glagoli. U daljem tekstu će biti detaljno opisano svako pravilo.

Kao prvo pravilo, a samim tim i prvi objekat koji se generiše jeste klasa Game. Ovo pravilo, predstavlja skup svih drugih pravila, i u njemu se nalaze svi neophodni elementi za igru. Naslov i uvod su obična string polja koja se ispisuju pri pokretanju igre. Sledeće polje klase Game jeste početak koji označava sa koje lokacije će se igra započeti. Korisnik mora izabrati neku od lokacija koje je pre toga definisao, a identifikuje je njenim nazivom.

Obeležje end je definisano pravilom GameEnd. Ono sadrži uslove (engl. *conditions*) i poruku u slučaju da je ispunjen uslov za kraj igre, koje je opisano pravilom Message. Za opis kraja igre su odabrana ova polja, jer kako će biti opisano u daljem tekstu, uslovima se mogu proveriti i stanja i lokacije, što je sasvim dovoljno da se postavi cilj koji treba postići u igri.

Pravilo Conditions sadrži listu uslova, koji su definisani pravilom Condition. Samo pravilo može biti ili StateCondition ili LocationCondition. Prva vrsta uslova se bazira na stanjima, koja će biti objašnjena kasnije detaljno, i proverava njegovu vrednost prostim logičkim poređenjem jednakosti. Dok pravilo provere uslova lokacije, proverava da li se objekat nalazi na određenoj lokaciji ili u određenom objektu. Za oba ova pravila je iskorišćeno referenciranje. Kako možemo primetiti u oba uslova postoji opcioni parametar, poruka. Ona je namenjena ako želimo da ispišemo određeni tekst u slučaju da taj uslov nije zadovoljen.

Pravilo Message definiše samu poruku, i njeno obležje jeste text, čija je vrednost tipa string. Za potrebe ovog pravila, odlučeno je da bude svedeno na minimum kako bi korisniku olakšali definisanje same igre.

Pravilo CID, prilagođava ID tip ovom jeziku, tako što sprečava korisnika da za ime izabere neku od rezervisanih reči koja se koristi u ostatku sistema. Ono je definisano pomoću negativnog predviđanja (engl. *negative lookahead*). Ovim putem se obezbeđuje da nijedna od rezervisanih reči ne može biti iskorišćena kao ID. Pravilo Keywords (ključne reči) definiše skup reči pomoću regex izraza. U rezervisane reči spadaju predefinisane akcije, lokacije, objekti, i reči korištene u ostatku jezike.

Opis je definisan pomoću pravila Description, koje u sebi sadrži samo obeležje tekst, koje je običan string.

Veze su opisane pravilom Connection. Ovo pravilo referencira početnu (engl. *from location*) i odredišnu (engl. *to location*) lokaciju, kao i smer, koji je određen pravilom Direction.

Smer (engl. *direction*) određuje smer kretanja, u odnosu na stranu sveta. Pravilo Direction je definisano kao uređena lista stringova.

Pravilo za objekat definiše njegov naziv (koji bi trebalo da bude jedinstven radi identifikovanja), opis i referencira njegovu lokaciju, koja može biti neka lokacija (definisana Location pravilom) ili može biti neki drugi objekat. Takođe sadrži skup uslova koji definišu šta sve mora biti ispunjeno kako bi taj objekat bio vidljiv, tj. da li će biti ispisivan pri ispisu lokacije i da li može da se istraži objekat. Više o samim akcijama i njihovoj konkretnoj implementaciji će biti kasnije u radu.

Sledeće pravilo omogućava da se odrede stanja. Stanje definišu ime, koje predstavlja identifikator, samo stanje, koje koristimo kao flag, i njegova vrednost može biti tačno ili netačno. Pored toga je neophodno definisati i prioritet, koji je tipa celog broja. Prioritet služi pri ispisu objekta, ako želimo da ispišemo neku poruku kada je stanje tačno, potrebno je da definišemo when_true poruku, ili u suprotnom when_false. Ove poruke nisu obavezne, ali je bitno napomenuti da će biti ispisana samo prva poruka za koju je zadovoljen uslov (iz tog razloga je bio potreban prioritet).

Postoje dve vrste akcija, jedne koje vrše neke promene - opisane pravilom StateAction ili one koje vrše opisivanje nekih objekata - opisane DescribeAction pravilom. Prvo će biti opisano StateAction pravilo. Njega definiše naziv, objekat/lokacija na koju se odnosi, skup uslova pod kojima može da se izvrši radnja, poruka u slučaju da je uspešno izvršena akcija i skup promena. Promene su definisane pravilom Change koje će biti opisano u narednom poglavlju. Druga vrsta akcija je opisana pravilom DescribeAction. One služe kao dodatak već ugrađenoj funkcionalnosti istraživanja (engl. *exam*). Prvo je potrebno definisati na koji objekat se odnosi, a zatim je potrebno definisati uslove, koji moraju biti ispunjeni kako bi se izvršila željena akcija. Pored toga je moguće definisati poruku i/ili print_condition, kojim se definiše da li se ispisuju objekti koji su sadržani u tom objektu koji se istražuje.

Izmene su definisane pravilom Change, koje mogu biti ili promene stanja (StateValueChange) ili mogu biti promene lokacije (LocationChange). Pri promeni stanja neophodno je referencirati neko stanje i dodeliti mu novu logičku vrednost. Pravilo za lokaciju je slično, neophodno je referencirati objekat i dodeliti mu novu vrednost za lokaciju, takođe referencirajući neku postojeću lokaciju ili objekat.

Poslednje pravilo definiše skup glagola, koji se koriste kao reference u pravilu StateAction. Njih definiše jedinstveno ime i opis.

6.2. Interpreter i parser

Kako za ovakav tip igara nije neophodno kreirati generator, kreiran je interpreter. On u vremenu izvršavanja kreira modele, dodatne metode i obrađuje zahteve koje igrač unosi. Kako bi funkcionalnosti samog igranja bile omogućene, klasa Game je proširena metodama koje to omogućavaju. Pored klase Game, definisana je i klasa GamePlay koja predstavlja naš parser i ona obrađuje unos od strane. Pored toga se iterira kroz listu glagola i kreiraju se metode kao što je malopre naznačeno. Ugrađene akcije su: *go, look, exam, inventory, new* i *exit*.

6.3. Generator

Druge komande koje je moguće izvršiti su *save* i *load*. Ove akcije predstavljaju proces čuvanja stanja igre na nekom medijumu, a load, akciju koja omogućava da igrač nastavi tamo gde je stao u trenutku kada je sačuvao. Za generisanje sačuvanog dokumenta koriste se Jinja šabloni.

Jinja predstavlja obrađivač šablona (engl. *template engine*) koji koristi semantiku sličnu Python – u i zasniva se na „bušenju rupa“ gde je potrebno kasnije umetnuti podatke. Za svaki šablon se proslede podaci koji se posle samo smeštaju na adekvatne pozicije. Iako većina logike treba da bude izmeštena van šablona, veliki deo posla i dalje može da se uradi unutar njih [8].

7. STUDIJA SLUČAJA

U daljem tekstu će biti prikazana jedna igra i opisani koraci koje je igrač učinio kako bi došao do cilja.

Nakon pokretanja igre, igrača prvo dočeka ekran na kome je ispisan naziv igre, i opcija da otpočne novu igru, ili nastavi prethodno sačuvanu. Za potrebe ove demonstracije biće otpočeta nova igra.

Na početku igre se ispiše uvodni tekst, a zatim sledi i opis trenutne lokacije. U prostoriji vidi kamen, seno i vrata od ćelije u kojoj se nalazi. Pored toga, iz prostorije postoji samo jedan izlaz. Prvo što igrač čini jeste da pokuša da izađe iz prostorije. Ovaj pokušaj je neuspešan i ispisuje mu se poruka da su vrata zaključana. Kako bi otkrio više o vratima on ih istražuje, a opis koji dobija mu kaže da mu je neophodan srebrni ključ. Zatim on pokušava da pomeri kamen, i ova akcija prolazi uspešno. Nakon toga, igrač odlučuje da ponovo osmotri sobu, nakon čega primećuje novi objekat, a to jeste srebrni ključ. Pre nego što otključa vrata, odlučuje da istraži i poslednji predmet u sobi a to jeste seno, koje daje beznačajan opis. Nakon uspešnog otključavanja vrata, igrač prelazi na novu lokaciju.

Pri ulasku u sledeću prostoriju, igrača dočekuje novi opis. Ovog puta, jedini objekat koji vidi jeste lampa i ima samo dva izlaza iz sobe, od čega je jedan kroz koji je ušao. Nakon istraživanja lampe, ne otkriva ništa specifično. Potom je uzima, i nastavlja u sledeću sobu.

Ovog puta, igrača dočekuje prostorija koja sadrži vrata, sanduk i ima dva izlaza iz nje. Igrač kao i u prvoj prostoriji pokušava da izađe, ali ne uspeva. Nakon istraživanja vrata, otkriva da mu je neophodan magični ključ. Zatim pokušava da otvori sanduk, ali je on zaključan. Nakon pokušaja otključavanja, otkriva da mu je potreban mesingani ključ. Nakon pretraživanja inventara, otkriva da jedino nije lampu iskoristio, nakon čega pokušava da je slomi. Ova akcija uspešno prolazi. Nakon istraživanja lampe, u njoj pronalazi mesingani ključ, i uzima ga. Zatim otključava i otvara sanduk, u kom pronalazi zlatni ključ. Da bi otključao vrata koja vode ka izlazu, mora da pronađe magični ključ. Kako nema više šta da istraži, pokušava da ga napravi. Ova akcija uspešno prolazi, uništava tri ključa koja je do sada skupio, i u inventar mu smešta magični ključ. Sa ovim ključem u inventaru, otključava vrata i napušta sobu. Pri napuštanju sobe, dobija poruku da je uspešno prešao igru.

8. ZAKLJUČAK

Kao i u svakom softverskom rešenju, tako i u ovom, uvek postoji prostor za napredak. Neke od ključnih stvari od kojih bi korisnici imali veliki benefit su:

1) **Mogućnost ulančavanja provera stanja ili lokacija u veći logički izraz**
Rezultat: omogućeno korišćenje I/ILI logičkih operatora

2) **Mogućnost definisanja sinonima za glagole**
Rezultat: veća fleksibilnost pri kreiranju igre

3) **Mogućnost definisanja osnovnog dela poruke, na koje bi se samo lepilo ime objekta, i ovakve poruke bi se vezivale za glagole**
Rezultat: manje linija koda, bio bi čitljiviji, a i kreatorima igara bi uštedelo vreme

4) **Mogućnost poboljšanja definisanja poruka kod uslova**

Treba napomenuti da i sa uvođenjem svih ovih izmena sama funkcionalnost ne bi bila drastično izmenjena.

9. LITERATURA

[1] Roger Firth i Sonja Kesserich, „The Inform Beginner's Guide“, The Interactive Fiction Library, August 2002

[2] Kevin Jackson-Mead i J. Robinson Wheeler, „IF Theory Reader“, April 2011

[3] https://alanif.se/downloads/documentation/manual.pdf (pristupljeno 10.10.2021.)

[4] https://adventuron.io/ (pristupljeno 10.10.2021.)

[5] I. Dejanović, „Prilog metodama brzog razvoja softvera na bazi proširivih jezičkih specifikacija. PhD thesis, Faculty of Technical Sciences, University of Novi Sad, January 2012”

[6] I. Dejanović, R. Vaderna, G. Milosavljević, Z. Vuković, „textX: A Python tool for Domain-Specific Languages Implementation“, Faculty of Technical Sciences, University of Novi Sad

[7] Dejanović I., Milosavljević G., Vaderna R.: Arpeggio: A flexible PEG parser for Python, Knowledge-Based Systems, 2016, 95, 71 - 74, doi:10.1016/j.knosys.2015.12.004

[8] https://jinja.palletsprojects.com/en/3.0.x/intro/ (pristupljeno 19.10.2021.)

Kratka biografija:

Aleksa Ivković rođen je u Novom Sadu 1997. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – računarstvo i automatika, odbranio je 2021.god.
kontakt: aleksa.ivk@gmail.com