Универзитет у Београду

Математички факултет

Мастер рад

Примена еволутивних алгоритама за процедурално генерисање садржаја у видео играма

студент: Филип Луковић 1048/2013

ментор: проф. др Владимир Филиповић

Београд, 2018

Ментор:

проф. др. Владимир Филиповић

Математички факултет

Универзитет у Београду

Чланови комисије:

проф. др. Филип Марић

Математички факултет

Универзитет у Београду

проф. др. Младен Николић

Математички факултет

Универзитет у Београду

Датум одбране:

_____

Резиме

Тема овог рада је примена еволутивних алгоритама за процедурално генерисање садржаја у видео игарама, а циљ рада је да покаже како се коришћењем еволутивних алгоритама процедурално генерисање садржаја у видео играма може унапредити.

Прво су у раду описане врсте садржаја у видео играма, након чега су описане технике за процедурално генерисање садржаја и изложене су њихове предности и мане. Као завршни корак описани су еволутивни алгоритми и њихова примена при генерисању садржаја у видео играма.

Цео процес је испраћен кроз имплементацију једне игре са неограниченим кретањем (енг. infinite runner) у којој је приказана употреба основних алгоритама за процедурално генерисање садржаја видео игре, а који су потом унапређени уз коришћење еволутивних алгоритама и других техника прилагођавања, како би се обезбедило да игра буде интерактивнија и прилагођена кориснику. Рад садржи значајне делове кода, као и поређење основне и унапређене верзије видео игре.

Видео игра имплементирана је у развојном окружењу Јунити (енг. Unity) и расположива је за довлачење као софтвер отвореног кода на адреси http://github.com/JupikeA/FruitStealer.

Кључне речи:

Процедурално генерисање, еволутивни алгоритми, видео игре

Садржај

1. Увод

Убрзан развој рачунарства и рачунарских система у последњих више од пола века прати и развој видео игара, тако да су оне у односу на 60-е године прошлог века, када је настала сама идеја видео игара, напредовале тако да су сада постале један од најпопуларнијих видова забаве на планети.

Видео игре су забавни или едукативни рачунарски програми који кориснику на основу његове интеракције са периферним уређајима приказују различит визуелни садржај. Ма колико комплексне или једноставне игре биле корисницима су битни што бољи визуелни доживљај и корисничко искуство, што је проузроковало непрестано такмичење у великој конкуренцији произвођача видео игара, а коначно довело до тога да је данас индустрија видео игара напредовала до веома високог нивоа. У бесконачној трци видео игара и хардверских ресурса које оне захтевају порасле су и саме могућности видео игара.

Виртуелна реалност (енг. Virtual reality) је постала саставни део видео игара, па као једина могућност за генерисања великих количина високо квалитетног садржаја какав је потребан данас је коришћење неког вида аутоматизације. Процес ручног креирања таквог садржаја захтева доста времена и велике тимове састављене од инжењера и уметника, што проузрокује да то буде неисплативо и да цена израде и дизајнирања једне видео игре буде као и прављење једног од холивудских филмова. Уз то, потребно је и да генерисани садржај буде што више налик стварном свету, да се тешко препознају шаблони по којима је генерисан или да шаблони у опште не постоје, што може бити изразито запажено у данас све популарнијим играма са неограниченим кретањем (енг. infinite runner).

Независно од комплексности или величине видео игара, један од најбољих и најбржих начин да се испуне жеље корисника истих у погледу велике количине квалитетног садржаја, као и да се видео игре учине занимљивијим је коришћење техника и алгоритама **вештачке интелигенције** (даље ВИ). Ипак, услед великих захтева за хардверсим ресурсима при коришћењу неких од алгоритама ВИ, потребно је поставити границу и уколико је могуће такве алгоритме користити само као последњи корак у низу, који би вршио најфинија подешавања и прилагођавања.

Генерисање садржаја у видео играма веома успешно се може вршити техникама за **процедурално генерисање садржаја** (даље ПГС). Сама чињеница да се технике за ПГС већ дужи низ година користе у рачунарској графици како би се генерисале реалистичне текстуре и објекти из природног окружења довољно говори, а све то је узроковано фкесибилношћу и ефикасношћу техника за процедурално генерисање садржаја. Технике за ПГС нису меморијски захтевне и могу се користити по потреби, односно садржај се не мора генерисати унапред и тако заузимати меморијске ресурсе, већ може бити генерисан по потреби, а уједно и прилагођен конкретној ситуацији или конкретном играчу како би корисничко искуство било додатно побољшано.

Процедуралне технике генерисања садржаја и поред свих наведених особина немају интелигентно понашање, већ се воде претходно дефинисаним алгоритмима и процедурама. У комбинацији са еволутивним алгоритмима, посебном врстом

алгоритама за вештачку интелигенцију, тај недостатак отклања и уједно добијаја добар начин за имитирање реалног света и биолошких процеса који су у великој мери заступљени у видео играма.

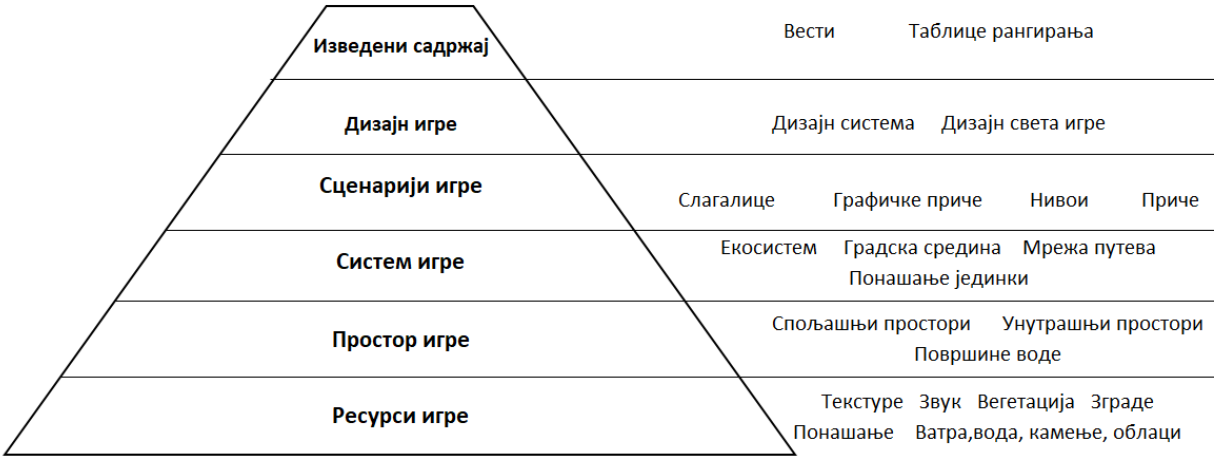
2. Врсте садржаја у видео играма

Пре описа конкретних техника за ПГС и примене ПГС у видео играма, важно је детаљније описати врсте садржаја у видео играма које се могу аутоматски генерисати уз помоћ алгоритама за ПГС.

Садржај видео игре може се поделити на следећих шест категорија:

- ресурси игре,
- простор игре,
- систем игре,
- сценарији игре,
- дизајн игре,
- изведени садржај,

где свака нижа категорија може бити генерисана уз помоћ генерисаног садржаја више категорије (простор видео игре може бити изграђен од генерисаних ресурса игре, а систем игре уз помоћ генерисаног простора игре и тако редом) [1].



Слика 1 - Типови садржаја у видео играма

2.1. Ресурси игре

Ресурси игре представљају основну јединицу садржаја видео игре. Појединачно, без контекста у који се касније постављају, нису ништа друго него независни објекти и као такви за корисника немају неког посебног значаја. У ресурсе игре убрајају се текстуре, звук, вегетација, зграде, понашање, ватра, вода, камење, облаци и сл.

- **Текстуре** су слике које се користе како би се геометрији и осталим објектима у игри додали детаљи који нису представљени самом геометријом или представили

материјали од којих су објекти заиста направљени. Текстурама се може симулирати дубина, односно трећа димензија. Оне се такође користе и за све делове корисничког интерфејса, како би се добио својеврсни визуелни печат и био препознатљив.

- **Звук** је једна од важнијих врста садржаја видео игара чији је примарни задатак да дочара тренутну атмосферу у игри. Такође, звук је важан повратни сигнал кориснику да се у игри нешто десило у односу на његову акцију.
- **Терен** заједно са **ватром**, **водом**, **камење**, **облацима** и другим ресурсима игре чини да видео игра изгледа што реалистичније и привлачније кориснику, али и доноси додатне погодности за корисника. У почетку ова врста садржаја искључиво служила за визуелни доживљај, али након великог пораста могућности видео игара могу служити као помоћни објекти у видео игри са којима корисник може интераговати.
- **Зграде** и остале грађевине су од суштинског значаја када је потребно представити градске средине у видео игри. Такође као и са вегетацијом, корисник са зградама може интераговати, а оне могу значајно мењати ток игре.
- **Понашање** је начин на који објекти у игри интерагују и са околином. Добро дефинисана понашања доприносе занимљивости, док процедурални начин генерисања даје игри велику дозу комплексности уз мали утрошак ресурса. Једном дефинисано, понашање може бити коришћено на различитим ресурсима или у различитим ситуацијама.

2.2. Простор игре

Простор игре је окружење у коме се ток игре дешава и делом се састоји од генерисаних ресурса игре. Примарна улога простора игре је да начином на који је састављен и саграђен усмери ток игре, а постојање објектата од којих је саграђен и везе међу истима учини значајним за корисника.

Простор игре се најчешће може поделити на **унутрашњи** и **спољашњи простор**, који се значајно разликују по структури, величини, нивоу детаља, начину интеракције и начинима генерисања истих.

- **Унутрашњи простори** су најчешће налик стварном окружењу у коме човек борави и углавном су подељени на мање целине попут соба (просторија). Собе могу бити просторно повезане, те се прелаз из једне у другу врши тачно одређеним путањама, али могу бити и потпуно одвојене па се прелаз врши помоћу разних замишљених портала. Распоред, величина и намена соба су од великог значаја за корисника. Генерисање распореда соба и објеката унутар њих може се успешно вршити аутоматизовано. Други тип унутрашњих простора су природне шупљине, колоквијално означене као пећине, код којих је сама геометрија веома изобличена и неправилна, а структура и повезаност се у многоме разликује од горе поменутих.
- **Спољашњи простори** одређени су конфигурацијом и структуром терена који представљају. Код оваквих простора, за разлику од унутрашњих који су ограничени, стиче се утисак непрегледности и бесконачности простора. Због

своје структуре генерисање спољашњег простора може се вршити применом техника са различитим нивоима аутоматизације од потпуно ручног до потпуно аутоматизованог генерисања. Делови терена се могу ручно правити од стране дизајнера, па касније састављати такође ручно или помоћу неког алата. Није редак случај да у једној видео игри поред спољашњег постоји и унутрашњи простор, а тада је од кључног значаја дефинисање начин преласка из једног у други.

- Поред наведена два, простор игре може се састојати од разних типова **водених површина**, попут река или језера, као и других великих природних или вештачких објеката или мањих објеката попут телепортационих портала.

2.3. Систем игре

Систем игре је најкомплекснија целина једне видео игре, па је за његову израду најчешће потребно користити веома сложене алгоритме и процесе, са сврхом да се помоћу тачно дефинисаних правила што реалистичније испрате процеси објеката у игри.

- У **природним екосистемима** систем игре за циљ има управљање распоредом, кретањем, растом, развојем и осталим природним процесима биљака и животиња, а у ретким случајевима и људи.
- Са друге стране у **градским срединама** циљ је да се поред природних, симулирају и друштвени процеси, комуникација и међусобна деловања људи, али и других објеката који су карактеристични за градску средину. У односу на сложеност простора игре који је представљен градске средине као један од основних елемената могу садржати велики број тачно распоређених зграда, које су међусобно повезане компликованом **мрежом путева** и остале инфраструктуре па је потребно симулирати и кретање великог броја превозних средстава и функционисање те компликоване инфраструктуре.

Највећи изазов је одређивање баланса како би све изгледало и функционисало као у реалном свету, а процедурални алгоритми овде дају доста добре резултате.

2.4. Сценарији игре

Сценарији игре су предефинисани токови које корисник може на лак начин пратити како би решио проблеме на које наиђе или како би извршио задатке коју му се постављају. Неки од најчешће употребљиваних сценарија при изради видео игара су слагалице, графичке приче, нивои и прича.

- **Слагалице** представљају проблеме које корисник решава на основу предходно стеченог искуства или логичким закључивањем у односу на стање окружења. Тежина решавања проблема зависи од нивоа предходно стеченог искуства или величине окружења у коме је проблем представљен.
- **Графичке приче** обично служе као смернице при дизајнирању видео игара, али се могу употребљавати и као средство за вођење корисника кроз ток игре.

Најчешће су налик стриповима представљени као секвенцијални низ сличица које описују кључне моменте или догађаје у видео игри.

- **Нивои** се користе у скоро свакој видео игри како би се целокупни ток игре раздвојио на добро дефинисане краће целине. Између нивоа се често убацују неки видови слагалица или графичких прича како би корисник упутио и припремио за следећу целину, а уједно могу послужити као место на коме се може сачувати тренутно стање у коме се видео игра налази.
- **Прича** је најчешће основа за креирање доброг корисничког искуства и средство које корисника води кроз игру. Прича би за разлику од слагалица, графичких прича и нивоа требало да буде јединствена на целокупном току видео игре и да унапред корисника упознаје са битним информацијама о догађајима који долазе, мада се некада дешава да у постоје и више прича па је кориснику остављен избор којим путем ће кренути.

2.5. Дизајн игре

Дизајн игре повезан је са свим категоријама садржаја описаним у предходним поглављима, а може се и рекурзивно повезивати са другим садржајима дизајна који се могу генерисати полу-аутоматизовано или уз помоћ разних специјализованих алата који дизајнерима омогућавају бржи развој.

Дизајн игре се састоји од **правила** која одређују скуп акција које корисник може извршавати у видео игри, као и **циљева** и **изазова** који одређују шта би корисник требало да испуни како би решио постављене проблеме, а осим тога садржи и **графичку тему** као важну естетску компоненту дизајна.

Као један од кључних процеса при креирању добре видео игре **дизајн система** видео игре за циљ има да се правила, циљеви и изазови равномерно распореде на све учеснике видео игре.

2.6. Изведени садржај

Изведени садржај у видео играма је онај садржај који је направљен као споредни садржај коришћењем информација из простора игре, сценарија игре и понашања корисника, а са циљем да побољшава корисничко искуство. Самим тим је пресудно да се у фокус кориснику поставе тачно одређене информације од значаја.

Најчешћи начин за приказ изведеног садржаја је путем **вести**, било путем новинских чланака, радија, телевизије или билборда. Осим тога, приказивање **таблице рангирања корисника** име веома позитивне ефекте како на тренутне тако и на потенцијалне кориснике.

3. Процедурално генерисања садржаја у видео играма

Процедурално генерисање садржаја (даље ПГС) је начин да се уз помоћ неког алгоритма или процедуре аутоматизовано креира велика количина садржаја [2]. ПГС налази примене у многим областима рачунарства и филмске уметности, али можда највећу примену проналази приликом генерисања рачунарске графике, звука и видео игара. Уместо да сав посао дизајнирања, цртања и моделовања ради човек, посао је препуштен рачунару, који за много краће време генерише велике количине високо квалитетног садржаја. Поред тога, ПГС за разлику од човека производи много разноврснији, а самим тим и занимљивији, садржај за крајњег корисника.

3.1. Историја

ПГС има дугачку историју у рачунарској графици као једној од области рачунарства. Процедуралне технике су годинама коришћене за генерисање реалистичних текстура и природних објеката и појава. Најзначајније технике које су коришћене су Перлинов шум (Ken Perlin), геометрија фрактала, Л-системи и граматика облика.

Историја видео игара показала је да овакав механизам може довести до успеха - већ 1980-те године је видео игра Роуг (енг. Rouge) отворила врата за ПГС у видео играма. Ова „графичка“ авантуристичка игра била је потпуно алгоритамски генерисана и сваки пут при игрању се генерисала нова авантура. Предходно описани тренд настављен је популарним серијом игара Диабло (енг. Diablo), као и игром Мајнкрафт (енг. Minecraft), које су само неке од познатих видео игара где се ПГС користи у великој мери [3].

Још један разлог због ког се ПГС показало као добро решење је то што ова техника није меморијски захтевна, а пре 30-так година меморија је био скуп ресурс. Исти проблем се јавио и касније, када је започет развој игара за мобилне уређаје, где се ова техника опет добро показала. Садржај генерисан помоћу техника за ПГС заузимао је 3 до 4 реда величине мање меморијског простора.

Дин Макри (Dean Macri) и Ким Палистер (Kim Pallister) су 2000-те године описали и представили прототип процедурално генерисаног тродимензионалног пејзажа са дрвећем и дводимензионалним облацима уз коришћење Перлиновог шума [1]. Већ 2001-е године Јоав Париш (Yoav Parish) и Паскал Милер (Pascal Mueller) су уз помоћ проширених Л-система направили су рачунарски програм “Сити Енџин” (енг. City Engine) који је служио за генерисање модела целога града. Систем је користио хијерархију правила како би генерисао шаблоне за улице и грађевине, али нажалост није подржавао генерисање геометрије зграда у реалном времену [1]. „Други Менхетн пројекат“ (енг. The other Manhattan nroject) из 2001. године описује алате за аутоматизовано генерисање градова налик на Менхетн уз помоћ статистичких параметара.

Полуаутоматски систем за реконструкцију ентеријера зграда од дводимензионалних архитектонских цртежа основа спратова, где су мапирање делова објекта и геометријске трансформације вршене аутоматизовано, али на основу корисниковог улаза, представљен је још 1998. године, а као крајњи резултат се добијао тродимензионални

модел. Гај Леки-Томпсон (Guy W. Lecky-Thompson) је 2001. године објаснило основе генерисања „бесконачног света“ у видео играма коришћењем случајно генерисаних бројева [1]. Александар Кристофер Волфганг (Christopher Wolfgang Alexander) је 1977. године дао смернице за конструкцију образаца за методичко креирање ентеријера и екстеријера градова, зграда, улица и башти са могућношћу одабира нивоа детаља помоћу образаца дизајна [1].

3.2. Подела техника за ПГС

Постоји много начина класификације садржаја насталог техникама за ПГС. Прво на основу чега се може направити разлика је то да ли је садржај **генерисан уживо** док је видео игра у току или је **изгенерисан у фази развоја** видео игре. Даље, садржај можемо разликовати на основу тога да ли се ради о **обавезном** или **необавезном** садржају. Генерисани садржај без којег играње видео игре није могуће је обавезан и мора бити одличног квалитета и финкционисати без грешака. Необавезан садржај је ту како би побољшао корисничко искуство и визуелни доживљај, односно понудио неке додатне могућности које не утичу на сам ток играња [4].

Степен контроле (енг. control degrees) означава флексибилност система, а зависи од броја улазних параметара и од тога у колико мери се њиховим мењањем мења и генерисани садржај. Системи који имају већи степен контроле могу произвести више различитог садржаја који може бити додатно прилагођен. Такође, уколико се за исте улазне параметре добија исти садржај могу се разликовати **стохастичко** и **детерминистичко** генерисање садржаја [4].

Када је реч о техникама које се користе приликом ПГС у видео играма прво се може извршити подела на следећих шест категорија (према којима ће ПГС и бити детаљније описане у наставку) [1]:

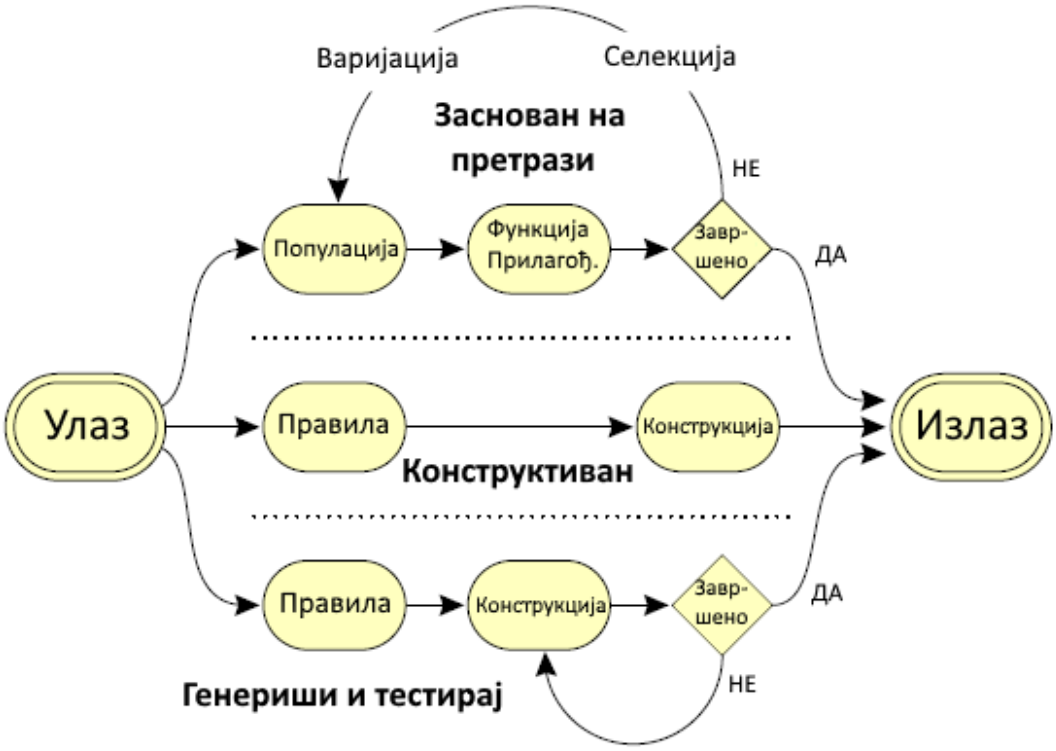
- генератор псеудо-случајних бројева (ГПСБ),
- генеративне граматике (ГГ),
- филтрирање слика (ФС),
- просторни алгоритми (ПА),
- моделовање и симулација сложених система (СС),
- вештачка интелигенција (ВИ).

Наредна подела може се извршити на основу начина процене квалитета изгенерисаног садржаја на основу чега се разикују следећи типови ПГС [5]:

- **засновано на претрази** (енг. Search Based) – садржај се генерише итеративно засновано на функцији прилагођености која усмерава претрагу,
- **конструктивно** (енг. Constructive) – садржај се генерише директно према одређеним правилима уз строгу проверу како генерисани садржај не би био „лош“,
- **генериши и тестирај** (енг. Generate & Test) - садржај се генерише директно према одређеним правилима и филтрира се према функцији прилагођености.

Уколико садржај није одговарајућег квалиета бива одбачен и поново се генерише нови садржај.

Технике засноване на претрази биће описане детаљније у поглављу 4.2.1.1 као један од начина за генерисање прилагођеног садржаја.



Слика 2 - Подела техника ПГС према начину процене квалитета садржаја

Тип алгоритма који ће бити коришћен за креирање конкретног елемента зависи од типа тог елемента. Различити алгоритми могу бити коришћени, почев од простијих као што су ГПСБ, преко ГГ, па све до најнапреднијих техника еволутивног програмирања и вештачких неуралних мрежа. Најчешће коришћене технике управо су ГПСБ и ГГ, као и СС, и биће детаљније описане. Такође, ГПСБ биће коришћени у имплементацији видео игре која је саставни део овога рада, док ће примена вештачке интелигенције за генерисање садржаја бити детаљније описана у посебном поглављу.

3.3. Генератори псеудо-случајних бројева (ГПСБ)

Генератор псеудо-случајних бројева (даље ГПСБ) је алгоритам за генерисање секвенци бројева који треба да апроксимирају особине секвенци случајних бројева, а кључне примене налази код симулација, методе Монте Карло (енг. Monte Carlo Method), видео игара и у криптографији [6].

Случајни бројеви представљају секвенце бројева који немају образац или формулу по којој се могу генерисати већ су настали случајно. Чак и за секвенце бројева који имају неки образац не може се рећи јесу ли настали случајно или управо по том обрасцу, тако да све секвенце бројева испољавају те „случајне“ особине. Овакве секвенце бројева су

значајне јер се управо оне јављају у природним појавама које је немогуће унапред предвидети. Добијање случајне секвенце бројева могуће је уз помоћ бацања новчића, мешања карата, бацање коцкица или рулета, међутим овакави начини генерисања секвенци случајних бројева су непрактични и превише спори.

Да би се наведене мане генерисања случајних бројева отклониле, конструисан је алгоритам за генерисање низа случајних бројева помоћу рачунара, али како је рачунар детерминистичка машина немогуће је применити концепт „случајног“. Услед те „мане“ рачунарских система конструисани алгоритам треба да што боље апроксимира генерисање низа случајних бројева, а уз помоћ малог броја улазних вредности.

ГПСБ почиње извршавање од произвољног **почетног стања** које можемо дефинисати параметром, а исто почетно стање увек производи идентичан секвенцу бројева која може бити и ручно израчуната. Свака секвенца ПСБ има неки **период** одређене дужине тако да се секвенце бројеви понављају на сваку дужине периода. Максимална дужина периода ГПСБ одређена је бројем стања, најчешће бројем битова, тако да дужина периода не може бити дужа од $2^n - 1$ где n представља дужину стања генератора изражено у битовима, али често може бити и краће [6].

Неки од најчешће коришћених техника за ГПСБ биће описане у тексту који следи.

3.3.1. Перлинов шум

Перлинов шум је врста математичке функција која се добија сабирањем више функција које су добијене случајним одабиром тачака где свака следећа функција има двоструко мању амплитуду и двоструко већу фреквенцију. Названа је по Кен Перлину (енг. Kenneth H. "Ken" Perlin) који је 1983. године развио овај алгоритам [6].

Алгоритам за Перлинов шум се најчешће имплементира помоћу дводимензионалне, тродимензионалне или четвородимензионалне функције, али је могућа имплементација за било који број димензија.

Алгоритам започиње **дефинисањем мреже** тако што се сваком чвору H-димензионе мреже додели случајно изабран јединични усмерени вектор H-димензионог простора. Како би одредили Перлинов шум за улазни H-димензиони усмерени вектор потребно је **израчунати скаларне производе** између улазног H-димензионог усмереног вектора и вектора чији је почетак конкретан чвор H-димензионе мреже, а крај унапред случајно изабрана тачка унутар простора који формирају чворови. Након тога је могуће одредити којем чвору мреже улазни вектор припада. Последњи корак је **интерполација** између 2^H добијених скаларних производа израчунатих у чворовима мреже како би се одредила конкретна вредност Перлиновог шума [7].

Највеће примене Перлиновог шума су у рачунарској графици као алгоритам за **градијентални шум**, односно стварање процедурално генерисаних градијенталних текстура ватре, дима, облака и још великог броја објеката из природе. Текстуре генерисане помоћу шума се могу лако мапирати на комплексне објекте, а имплементација шума је једноставна и налази се у већини програмабилних рутина за

сенчење(енг. shaders) и графичких картица попут Енвидије (енг. NVIDIA), тако да захтева мале меморијске ресурсе.

Перлинов шум примене налази и у генерисању и обради звукова за видео игре, а у комбинацији са другим техникама ФС, ПА и СС користи се за генерисања комплетних мапа спољашњег света.

3.3.2. Линеарни конгруентни генератор

Линеарни конгруентни генератор (даље ЛКГ) представља једну од најстаријих и најпознатијих метода коришћених за ГПСБ, а извршава се коришћењем линеарне функције над пољем реалних бројева која није непрекидна у свим тачкама тог поља. Логику која стоји иза ЛКГ је лака за разумевање и имплементацију, посебно због чињенице да се користи аритметика по модулу која је хардверски подржана у рачунару [6].

ЛКГ је дефинисан следећом рекурзивном функцијом

$$X_{n+1} = (aX_n + c) \, mod \, m$$

где X представља секвенцу случајно генерисаних бројева, а

$$m, \quad 0 < m - \text{модуо}$$

$$a, \quad 0 < a < m - \text{множилац}$$

$$c, \quad 0 \leq c < m - \text{увећање}$$

$$x_0, \quad 0 \leq x_0 < m - \text{почетна вредност}$$

Предности ЛКГ су брзина и мали меморијски захтеви и уз коришћење добрих параметара може се произвести униформна расподела генерисаних бројева, али не би требало да се користи у апликацијама које захтевају висок ниво случајности при генерисању секвенци случајних бројева.

Кључни проблем који се јавља код једноставног линеарног алгоритма је то што је период ограничен модулом по коме се рачуна. Коришћењем **успореног Фибоначијебог генератора** (енг. Lagged Fibonacci generator) добијају се знатно већи периоди. Основа алгоритма се заснива на Фибоначијевом низу помоћу формуле

$$S_{i+1} = S_{i-p} \pm S_{i-q}$$

па услед коришћења више предходних вредности може доћи до понављања бројева у оквиру једне секвенце [6].

3.3.3. Линеарни повратни померачки регистар

Линеарни повратни померачки регистри (даље ЛППР) су померачки регистри чији се улазни бит рачуна као линеарна функција предходног стања регистра, а представљени су као бинарни низ одређене дужине. Као функција се најчешће користи екскузивно или

(енг. eXclusive OR, XOR) и резултат се смешта у крајњи леви бит, а остатак се помера на десну страну. Бит који је „излетео“ на десно је управо следећи случајни бит излаза. Предност ЛППР то што се лако може додати ентропија у систем само додавањем нових информација у функцију, док је период без додавања ентропије једнак $2^n - 1$ где је n број битова у регистру [6].

Поред могућности додавања ентропије у систем, предности су и веома велика брзина, прилично добре случајне секвенце и дугачак период.

3.3.4. Метода са средњим квадратом

Први алгоритам за ГПСБ путем рачунара, осмишљен од стране Џона фон Нојмана (John von Neumann) 1946. Године, био је познат као **Метода са средином квадрата** (енг. middle-square method) [6]. Алгоритам је текао тако што се узме се било који број, број се квадрира, уклоне се средње цифре добијеног броја и те цифре представљају случајан број и семе за следећу итерацију. Због евентуалног брзог понављања секвенци овај алгоритам је после неког времена замењен сложенијим алгоритмима.

3.3.5. Примене

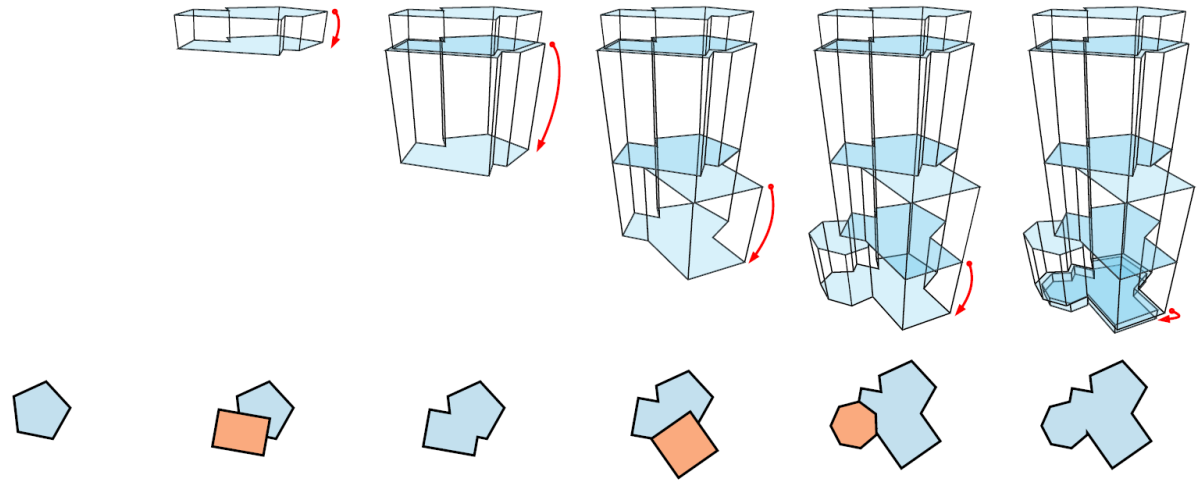
Технике за ГПСБ могу се комбиновати са напредним алгоритмима за претрагу простора и уз помоћ разних параметара конструисати унутрашњи простор који задовољава дефинисана правила.

Већине описаних ГПСБ показују нека лоша својства и ограничења која проузрокују лошу апроксимацију генерисања секвенци случајних бројева, а она укључују:

- периода може бити краћа од очекиване за нека почетне вредности,
- лако предвиђање свих чланова секвенце када је позната почетна вредност,
- недостатак униформости дистрибуције великог броја генерисаних бројева,
- Корелација узастопних вредности,
- слаба димензиона расподела излазних секвенци,
- различите дужине дистрибуције између одређених вредности од оних који се дистрибуирају у случајним секвенцама.

Коришћењем псеудослучајности и вероватноће може да се направи утисак неизвесности неког догађаја и да се понашање учини непредвидивим [8].

Ова техника примењује се у свим жанровима видео игара и приликом генерисања скоро свих врста садржаја у видео играма, али главна примена ГПСБ је за генерисање текстура на мапама и објектима, генерисање звукова и осталих ресурса игре. Као што је рецимо у природи немогуће одредити функцију по којој су и на којем месту настала природна испупчења или удубљења, тако се ни у видео играма не треба бавити тиме, већ то треба препустити некој од техника за ГПСБ и евентуално ручно извршити проверу резултата. Такође ни генерисање расподеле ресурса или особина ликова у видео играма ништа боље неће бити обављено од стране човека него од стране ГПСБ.



Слика 3 - Визуелни приказ генерисања зграде коришћењем техника ГПСБ

Приликом доношења одлука тако да постоји више добрих решења или су сва решења довољно добра или пак нема довољно података за доношење одлуке ГПСБ представљају идеалан начин доласка до решења, а као производ уносе и дозу непредвидивости. ГПСБ такође се примењују и у ситуацијама када је потребно покрити неке аномалије или случајеве код којих се неки процес или догађај дешава са одређеном вероватноћом пошто су и сами засновани на вероватноћи.

Веома важна карактеристика ГПСБ је да коју год од горе наведених примена имали не захтевају велики број улазних података за одређивање исправног решења. У великом број случајева довољно је задати само опсег вредности вројева које желимо да добијамо. За напредније технике можемо дефинисати и додатне параметре, евентуално расподелу вредности како би на излазне вредности био извршен већи утицај, али све се ради параметарски тако да је то меморијски незахтевно и лако се преноси преко мреже, а то је веома битно у данашње време када скоро и не постоје видео игре које нису на неки начин повезане преко интернета.

3.4. Генеративне граматике (ГГ)

Појам **генеративне граматике** (даље ГГ) још 50-их година прошлог века описао амерички лингвиста **Чомски** (Avram Noam Chomsky). Углавном их је користио за проучавање синтаксе лингвистичких фраза.

ГГ представљају тип формалних граматика и састоје се од скупа симбола датог језика и унапред дефинисаног скупа правила преписивања чијом рекурзивном применом је у теорији могуће генерисати било коју могућу реченицу језика дефинисаног том граматиком. Заснивају се на рекурзивном процесу преписивања (замене) почетног скупа симбола помоћу скупа правила преписивања. Описани концепт може бити примењен и

на друге дисциплине попут процедуралног генерисања велике количине садржаја у видео играма [9].

Граматике се разликују по начину примене правила преписивања, тако да **контекстно слободна граматика** је она граматика код које се свако продукцијско правило односи само на један конкретан симбол, а не и на његове суседе. Уколико примена правила зависи и од суседа конкретног симбола, ради се о **контекстно осетљивој граматици**. Неке од контекстно слободних граматика су Л-системи и граматика облика.

Надаље, граматике се могу разликовати и према броју правила замене за један конкретан симбол. Када постоји тачно једно правило за сваки симбол, ради се о **детерминистичкој** граматици, а уколико постоји више правила замене и свако се бира према одређеној вероватноћи ради се о **стохастичкој** граматици, код које се за избор тачног правила преписивања у великом броју случајева користи нека од техника за ГПСБ.

3.4.1. Линденмајерови системи

Л-системи или **Линденмајерови системи** (Lindenmayer systems) представљају тип ГГ која се углавном користи за процедурално генерисање садржаја, а које је развио и широј јавности представио мађарски теоретски биолог и ботаничар Аристид Линдермајер (енг. Aristid Lindenmayer) по коме и носе име [9].

Л-системи могу бити дефинисани уз помоћ скупа

$$G = \{V, S, \omega, P\}$$

где је

V - скуп симбола који садрже елементе који могу бити замењени (нетерминали)

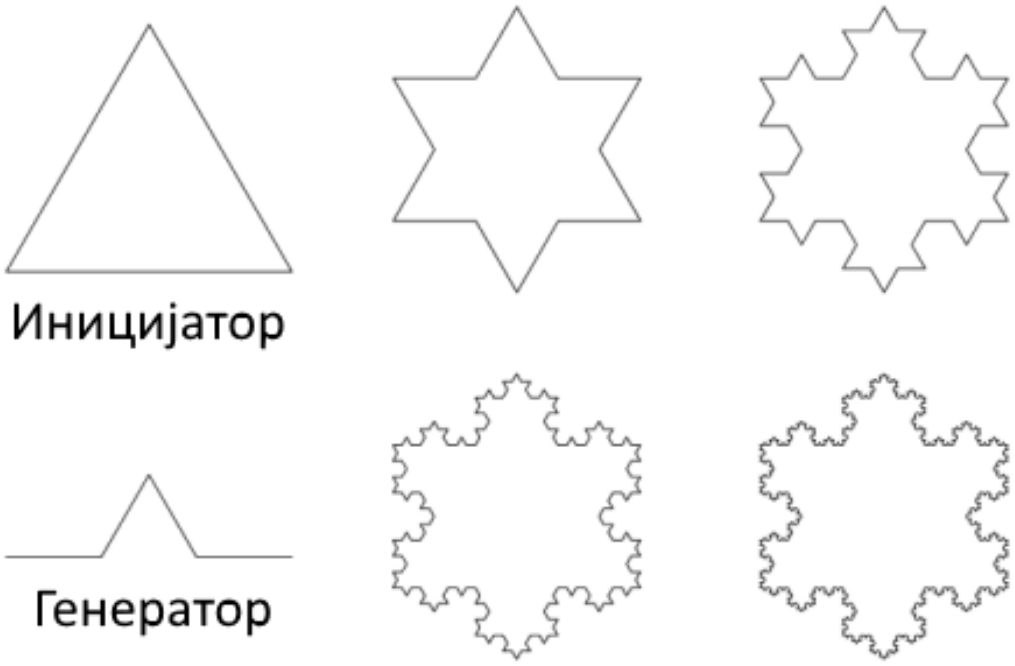
S - скуп симбола који садрже елементе који остају непромењиви (терминали)

ω - низ симбола који дефинишу почетно стање система

P - скуп правила извођења или извођења која дефинишу начин на који нетерминали могу бити замењени терминалима или другим нетерминалима. Продукција се састоји од два стринга, претходника и следбеника

Важно својство Л-система које их разликује од осталих сличних граматика је то што се замена симбола може вршити упоредо и током једне итерације применити више правила замене. Ово својство одређује Л-системе као строги подскуп језика, док би се применом правила једно по једно створио само језик [9].

Систем се зауставља када се достигне одређени број итерација, када нема више симбола који могу бити замењени или када је корисник задовољан тренутним резултатима.



Слика 4 - Пример генерисања пахуље уз помоћ генеративне граматике

3.4.2. Граматика облика и граматика раздвајања

Граматика облика (енг. shape grammar, даље ГО) је посебан тип контекстно слободне формалне граматике. Почетак ГО налази се још у радовима Албертија (Leon Battista Alberti) и Витрувиуса (Marcus Vitruvius Pollio) у којима су описани параметри и њихове варијације за коришћење у архитектури (обично храмова) као скуп параметризованих правила. Касније ГО почиње да буде коришћена у рачунарској графици, као скуп правила за генерисање скупова зграда за потребе видео игара и рачунарске анимације, а након тога и за реконструкцију или стварање потпуно нових зграда [8] [10] .

ГО бар делимично је зависна од контекста у коме се налази, а разлог томе лежи у чињеници да пре него што неко од правила буде примењено потребно је размотрити услове и евентуално извршити нека израчунавања, што је чини тежом за имплементацију од осталих чисто контексно слободних граматика. Али са друге стране и могућности ГО су далеко веће. Применом ГО уз помоћ примитивних облика уобичајено се добија само „груба слика“ зграде на коју се касније додају детаљи попут боје, прозора, врата и осталих елемената архитектуре.

3.4.3. Примене

Л-системи су првобитно били предложени као математичка теорија за моделовање раста и развоја биљака што је и сада када се примењују у разним областима рачунарске науке, поред моделовања раста и развоја других живих организама, највећа примена Л-система. Због своје рекурзивне структуре налик фракталима ова техника је идеална за моделовање јединки биљака које су сличне једне другима, обично дрволиких биљака,

чији делови су представљени симболима. Технике ГГ које користе графове повезаности се могу користити за моделовање веома сложених система вегетације.

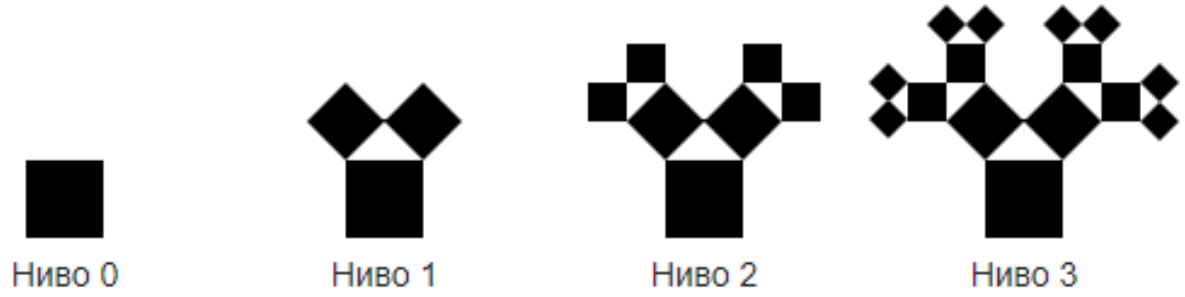
Коришћењем техника ГГ попут „побољшаних“ Л-система може се изгенерисати велики број зграда, кућа и осталих грађевина користећи релативно мали број унапред дефинисаних правила или садржаја направљеног од стране корисника. Зграде добијене малим бројем правила не могу имати превише компликовану структуру, али коришћењем већег броја правила и њиховим усложнавањем могу се добити прилично комплексне зграде. Али чак и након уноса великог броја параметара потребно је генерисани садржај проверити визуелно, јер неки резултати могу бити неодговарајући онима који се виђају у стварном свету.

Унутрашњост зграда и других грађевина може се генерисати комбинацијом техника које припадају ГГ и ГПСБ. Технике засноване на ГГ користе се за генерисање графа повезаности просторија или других мапа, док се технике за ГПСБ користе за генерисање самих просторија. Технике за ГПСБ могу се комбиновати са напредним алгоритмима за претрагу простора и уз помоћ разних параметара конструисати унутрашњи простор који задовољава дефинисана правила.

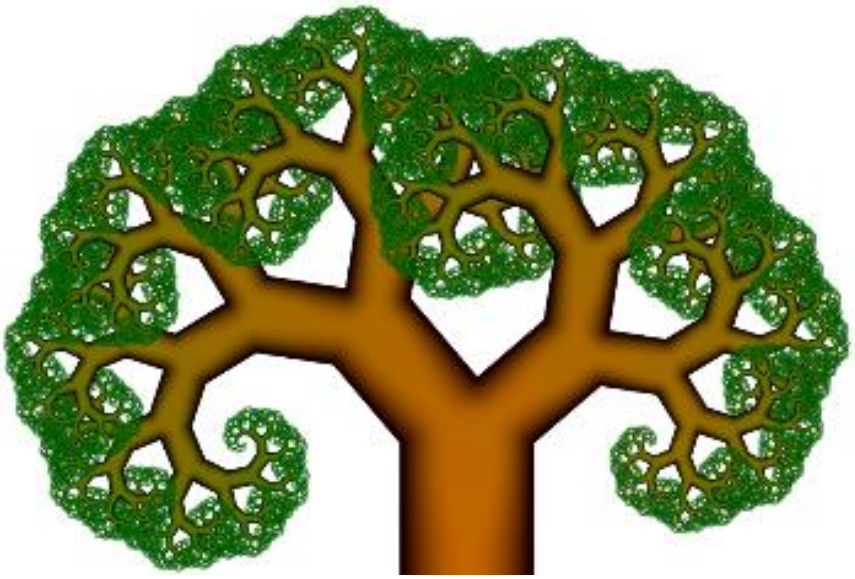
Л-системи се такође могу користити за генерисање мреже путева унутар градски средина дефинисањем кључних параметара и за моделовање једноставнијих понашања објеката, док се контекстно осетљиви Л-системи најчешће користе за моделовање сложеног понашања објеката попут експлозије ватромета без потпуног физичког модела.

Процедурално генерисање звука може се извести коришћењем техника ГГ или било којег другог система заснованог на правилима која дефинише композитор или моделар звука. Предност оваквог начина генерисања звукова у видео играм су мали меморијски захтеви и флексибилност у односу на минималне почетне ресурсе. Мана је то што је предходно потребно преточити секвенце звука дефинисане нотама у правила или природни језик, али се коришћењем ВИ овај посао може динамички извршавати.

Као добар пример коришћења ове технике на сликама испод приказано је **Питагорино дрво**, фрактал у две димензије (по потреби он може бити конструисан и у 3Д) настао од квадрата и правоуглих троуглова (добио назив по Питагориној теореми) [11]. Конструкција дрвета започиње квадратом, над којим се конструишу два нова квадрата (над катетама једнакокраког правоуглог троугла чија је хипотенуза једна од страница почетног квадрата). Процес се наставља итеративно, тако да се у сваком корако додаје по 2^n нових квадрата укупне површине као почетни квадрат. Троуглови који се конструишу над почетним квадратима не морају бити једнакокраки, а чак не морају бити ни правоугли па се променом углова може добити интересантно дрвеће.



Слика 5 - Питагорино дрво (почетне итерације)



Слика 6 - Обојено питагорино дрво

3.5. Филтрирање слика (ФС)

Филтрирање слика (енг. image filtering, даље ФС) је процес измене или побољшавања слике применом функција за филтрирање. Процесом ФС могу се нагласити једне, а уклонити друге карактеристике слике применом замућивања (енг. blurring), равњања (енг. smoothing), изоштравање (енг. sharpening), побољшања ивица (енг. edge enhancement) или неких других филтера. Значајни типови ФС су конвулација, корелација и бинарна морфологија.

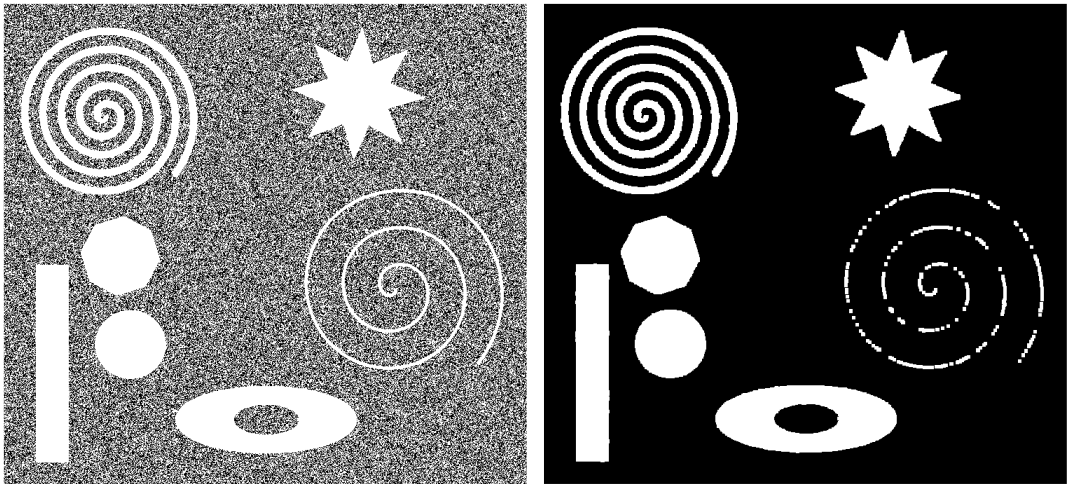
ФС је операција суседства (енг. neighborhood operation) која вредност било ког пиксела у излазној слици одређује примењујући одређени алгоритам на вредности пиксела суседних у односу на одговарајући пиксел који се обрађује, а **линеарно филтрирање** је тип филтрирања код кога се излазна вредност одређеног пиксела израчунава као линеарна комбинација вредности суседних пиксела.

Типови линеарног ФС код којега се вредност сваког излазног пиксела израчунава као тежинска сума вредности суседних пиксела називају се **конволуција** (енг. convolution) или **корелација** (енг. correlation), које су веома сличне. Филтрирање се врши уз помоћ

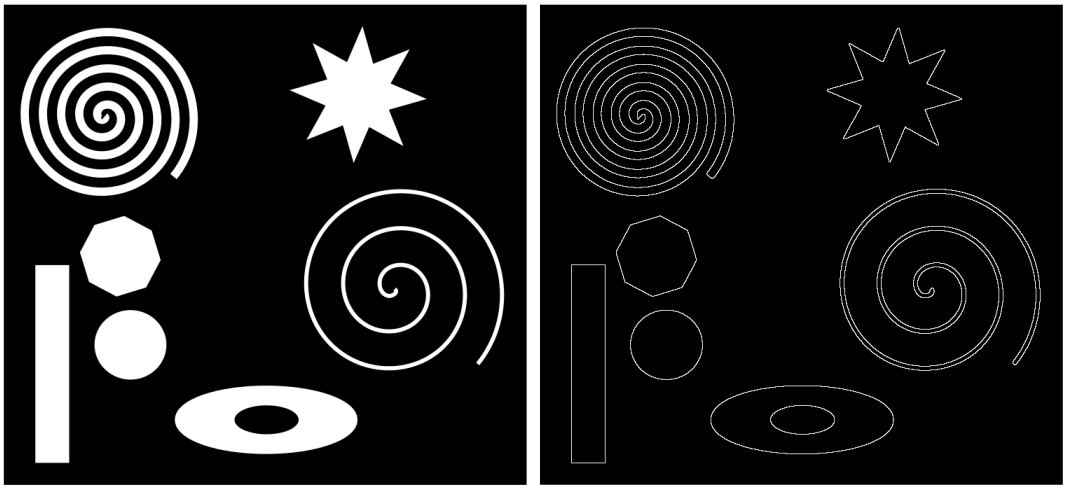
дводимензионалних функција редом названих **конволуциона (корелациона) матрица** или филтер. Овакви типови филтера су веома ефикасни када се примењују директно на графичкој карти која има велики број језгара, пошто су картие погодне за рад са матрицама.

Бинарна морфологија (енг. binary morphology) је нелинеарни тип филтрирања слика налик на конволуцију намењен бинарним сликама (дводимензионална матрица која садржи једино 0 и 1). Могу се разликовати **ерозија** и **дилатације**.

Текстурисање засновано на шаблонима користи се како би се имитирао висок ниво детаља као код текстура високих резолуција, а конволуциони филтери попут брушења и равњања могу динамички унапрадити текстуре у видео играма. У комбинацији са другим техникама ФС се користи при генерисању висинских мапа, као и за ерозију и/или изравњавање терена.



Слика 7 - уклањање шума са бинарне слике



Слика 8 - детекција ивица бинарне слике

3.6. Просторни алгоритми (ПА)

Најчешће коришћени ПА су поплочавање и раслојавање (енг. tiling and layering), правило коначне поделе простора (енг. grid subdivision), фрактали (енг. fractals) и воронејеви дијаграми (енг. Voronoi diagrams).

3.6.1. Поплочавање и раслојавање

Техника **поплочавања** највећу примену има приликом текстурисања. Заснива се на текстурисању великих површина помоћу текстура или шаблона малих димензија тако што се шаблони слажу један до другог као керамичке плочице, па отуда и назив технике. Поред предности као што су меморијска ефикасност и могућност поновног коришћења шаблона независно о димензијама објекта који се текстурише, мана је то што су шаблони лако видљиви уколико нису припремљени за поплочавање (Слика 9). Овај проблем може бити решен додавањем шума на текстуру коришћењем неких других техника за ПГС.

Слика 9 - Пример поплочавања

Раслојавање је техника постављања једноставних шаблона једних преко других чиме се добија сложенији шаблон. Како би ова техника пружила већу флексибилност процес спајања дефинисан је функцијом спајања [12].

Због минималних меморијских захтева и брзине извршавања поплочавања и раслојавања користе се у многим комерцијалним видео играма (највише у стратегијама у реалном времену). Ове две технике могу се користити у комбинацији.

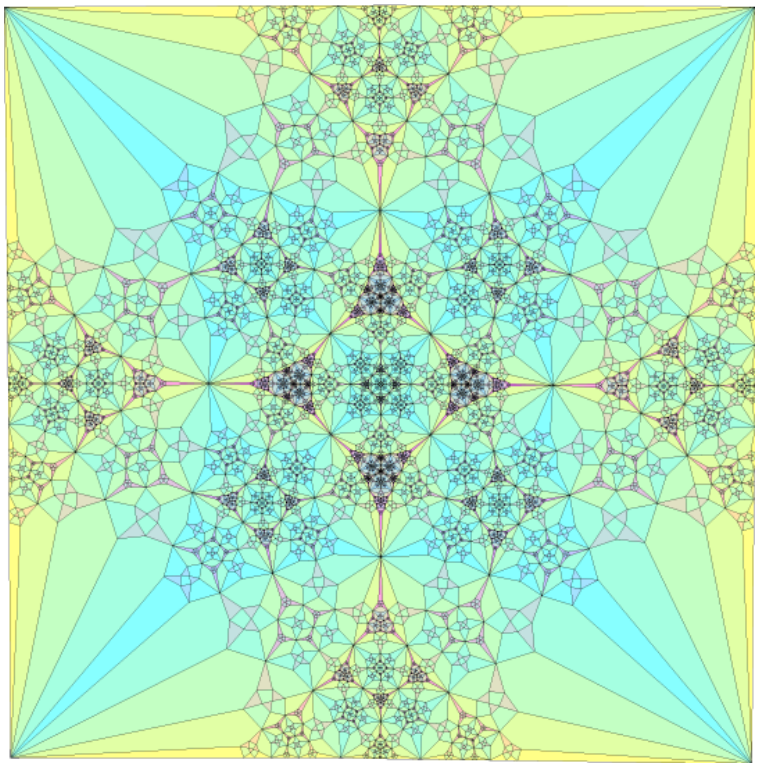
3.6.2. Фрактали и правило коначне поделе простора

Фрактали су рекурзивни геометријски облици чији су сви делови слични целини. Неке од значајних особина фрактала су [12]:

- показују фину структуру независно од степена увећања,
- сами су себи слични,

- имају једноставну и рекурзивну дефиницију.

Правило коначне поделе простора је рекурзивни начин поделе полигона на све мање и мање делове, тако да се добијају облици веома слични фракталима (као што се може видети на Слика 10).



Слика 10 - Фрактал настао помоћу коначне поделе простора

Ове две технике обично се користе у рачунарској графици за генерисање слика које представљају природне објекте, као и за генерисање геометријских текстура. Предност ових техника је то што се уз мали утрошак меморије могу бити изгенерисане неограничено велике текстуре које задржавају своја својства приликом увећања.

Фрактали се такође користе да опонашају раст биљака и природне процесе попут ерозије, а такође се и неки од алгоритама за генерисање пећина заснивају на фракталима. Већ је поменуто да Л-системи поседују фрактална својства.

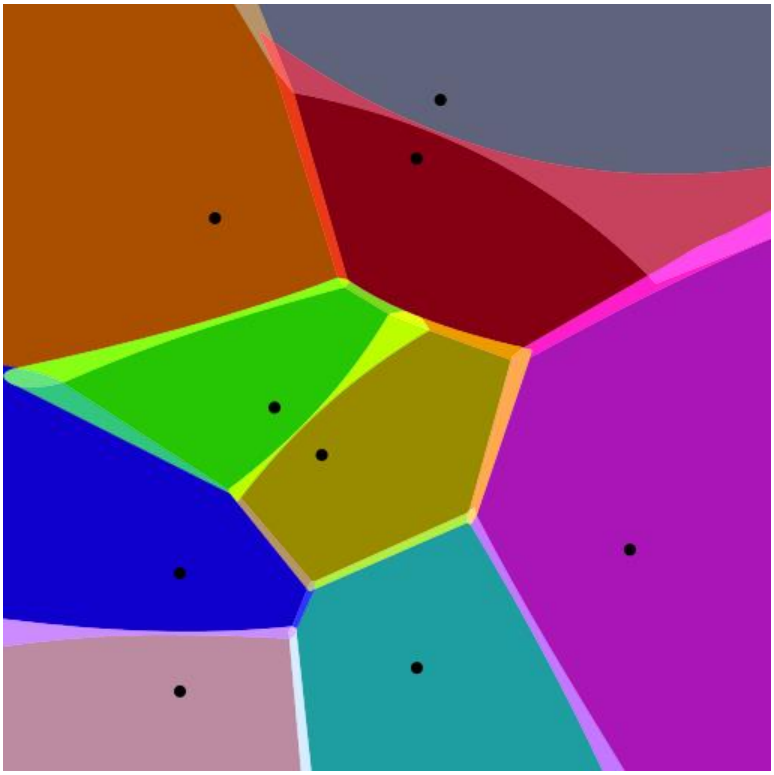
Коришћењем фрактала за генерисање терена могу се добити веродостојне планине и узвишења. Алгоритам **дијамант-квадрат** који се најчешће користи за висинске мапе заснива се на итеративној подели простора и додељивањем случајно одабраних висина средиштима тако подељеног простора. Овај тип алгоритам може се услед своје брзине користити за генерисање терена у реалном времену [11].

3.6.3. Воронејов дијаграм

Воронејов диаграм (енг. Voronoi diagrams) представља геометријски алгоритам који служи за поделу простора. Сврха је да се простор подели тако да свака тачка (објекат

или јединица) буду најмање удаљени од дела коме припадају у односу на друге делове простора (што се може видети на).

Најчешћа примена Ворнојевих дијаграма у видео играма је при генерисању мрежа путева и аутопутева при изградњи градова као спољашњих мапа. Такође се примењују и за одабир локације за генерисање неких важних објеката, попут болница, школа или супермаркета, који је потребно да буду једнако удаљени од осталих објеката на том простору. У комбинацији са другим техникама могу служити за генерисање читавих простора природних екосистема.



Слика 11 - Илустрација Воронејових дијаграма

3.7. Моделовање и симулација сложених система (CC)

Сложени системи су они системи који показују неке или чак могуће све од следећих карактеристика [13]:

- повратне спреге,
- неки степен спонтаног поретка,
- робусност поретка,
- појаву организације,
- бројност,
- хијерархијску структуру.

Сложени систем најчешће је систем састављен од великог броја различитих компонената које интереагују, а овакви системи су чести у природним екосистемима,

као и у урбаним срединама. Могу се разликовати ћелијски аутомати, тензорска поља, симулације засноване на агентима и други мање коришћени СС.

3.7.1. Ћелијски аутомат

Ћелијски аутомат (енг. Cellular automaton, даље ЋА) је дискретни модел система **ћелија** са следећим карактеристикама [14]:

- састоји се од правилне **мреже** ћелија било које коначне димензије,
- Свака ћелија има неко од коначног броја **стања,**
- свака ћелија има тачно дефинисане **суседне** ћелије. Њихов одабир се може вршити на било који начин, али обично су то заиста суседне ћелије у мрежи,
- **почетно стање** дефинисано је стањем сваке од ћелија унутар мреже,
- **нова генерација** се ствара према неком фиксном правилу које одређује ново стање ћелије у зависности од тренутног стања и стања ћелија у њеном суседству.

Према британском научнику Стивену Волфраму (Stephen Wolfram) ЋА можемо према сложености самих правила поделити на четири класе [14]:

- **класа 1** - Скоро сви почетни обрасци развијају брзо, стабилно и хомогено стање. Свака случајност у почетном образцу нестаје.
- **класа 2** - Скоро сви почетни обрасци развијају брзе стабилне или осциловане конструкције. Неки од случајности у почетном обрасцу се могу филтрирати. Локалне промене у почетном обрасцу теже да остану локалне.
- **класа 3** - Скоро сви почетни обрасци се развијају псеудо-случајно или на хаотичан начин. Све стабилне структуре које се појављују се брзо уништавају од стране околне буке. Локалне промене у почетном обрасцу имају тенденцију да се шире на неодређено време.
- **класа 4** - Скоро сви почетни обрасци се развијају у структуре које реагују на сложене и интересантне начине, уз формирање локалних структура које су у стању да преживе дуже време.

ЋА који показају сличне особине као и разни системи из стварног живота, укључујући биолошке и хемијске процесе, могу се користити за њихову симулацију и моделовање физичке реалности.

Генерисање унутрашњих природних простора попут пећина могуће је вршити применом широког спектра алгоритама, обично заснованих на техникама за ГПСБ и СС. Технике које припадају ГПСБ користе се за генерисање лавиринata и соба у мањим пећинама, док се за генерисање великих и компликованих пећина обично користите ЋА. Они могу бити коришћени и за генерисање биљака у земљишту, као и термалну или хидро ерозију [15]. Кретање животиња као и у стварном свету требало би да буде **случајно кретање** које се може симулирати помоћу ЋА.

Ова техника може бити коришћена како би се генерисао звук различитог нивоа детаља користећи ефекат слабљења и симулирало ширење звука кроз ваздух или течности зарад стварања реалистичне атмосфере током играња.

3.7.2. Симулација заснована на агентима

Под термином **интелигентног агента** подразумева се хардверски или много чешће софтверски систем који задовољава следеће карактеристике [16]:

- поседује аутономни начин рада без потребе за акцијама корисника,
- у могућности је да интерагује и комуницира са осталим агентима из окружења,
- реагује на промене које се дешавају у њиховом окружењу,
- има способност да преузме иницијативу за одређене акције.

Симулација заснована на агентима (даље СЗНА) у видео играма најчешће се примењују за моделовање рачунарски вођених ликова који задовољавају све карактеристике наведене изнад, али поред тога се примењује и за моделовање спољашњег простора и физичке реаности.

СЗНА се користе како би се симулирало понашање велике групе људи или животиња и како би понашање сваке јединке из групе било аутономно. Овим начином симулације олакшава се тежак посао ручне израде анимација понашања. СЗНА обично користе више врста агената, прве **физичке агенте** који симулирају физичко понашање, скелет или сензоре каракрера, а друге **когнитивне агенте** за размишљање, доношење одлука, планирање и учење.

Повећањем сложености и графичке привлачности видео игара расту и очекивања играча. Очекивања се своде на све веће интераговање међу објектима унутар видео игре, као и побољшање физичке реалности. Под побољшањем физичке реалности мисли се на то да је у што већој мери потребно испоштовати физичке законе и начин на који објекти унутар видео игре реагују на акције играча. СЗНА представља право решење за преходно поменута очекивања играча, а при употреби не захтева превише рачунарских ресурса и може се извршавати у реалном времену.

Поред моделовања рачунарски вођених ликова СЗНА се може користити и за генерисање спољашњег простора у коме се игра одвија. Генерисање терена помоћу агената заснива се на томе да се почетни празан терен обрађује уз помоћ великог броја агената који му дају коначан облик. Обрада терена извршава се у фазама и сваки агент је задужен за тачно одређени сегмент обраде. Овакав начин генерисања терена има много већи степен контроле од генерисања терена заснованог на фракталима [11]. Сличан начин генерисања се може применити и на генерисање реалистичних текстура.

Супротно предходно наведеном, за генерисање мреже путева обично је довољно користити само два типа агената, један за истраживање терена и генерисање путева, а други за повезивање путева једних са другим како би се креирала мрежа [1].

4. Вештачка интелигенција

Живи организми као што су животиње и људи имају посебну врсту интелигенције која им помаже у доношењу одређене одлуке да нештоураде. Рачунари су са друге стране само електронски уређаји који примају податке, великом брзином извршавају логичке и математичке операције и избацују резултате. **Вештачка интелигенција** (даље ВИ) у суштини представља начин за стварање система способних да приликом обављања специфичних операција доносе одлуке као живи организми [17].

Приликом истраживања закона природе у сврху науке или израде корисних производа у инжењерству често се наилази на проблеме, од којих су оптимизациони проблем и проблем учења једни од најзначајнијих. ВИ као једна од најновијих поља у науци и инжењерству покушава управо да реши проблеме оптимизације и учења код рачунара. Примена ВИ је веома широка и у зависности од конкретне примене постоје различите имплементације алгоритама и дефиниције ВИ. Области у којима се примењује ВИ могу се поделити на [13]:

- обрада природног језика (енг. natural language processing),
- представљање знања и разумевање (енг. knowledge representation and reasoning),
- планирање (енг. planning and scheduling),
- претрага (енг. search methodologies),
- методи управљања (енг. control methods),
- филозофске основе вештачке интелигенције (енг. philosophical/theoretical foundations of artificial intelligence),
- дистрибуирана вештачка интелигенција (енг. distributed artificial intelligence),
- рачунарски вид (енг. computer vision).

4.1. Еволутивни алгоритми

Еволутивни алгоритми (даље ЕА) чине групу алгоритма који обављају задатке оптимизације и/или учења са могућношћу побољшања и поседују бар следеће кључне карактеристике [18]:

- јединке су груписане у популацију која је основ еволутивног процеса и даје могућност паралелног решавања проблема учења или проблема оптимизације,
- селекција се врши на основу вредности прилагођености јединки и то природно проузрокује пораст оцене целокупне популације што чини темељ оптимзације,
- над јединкама се примењују оператори мењања над генима и тако се постиже претерага простора решења.

Оптимизациони проблем састоји се у проналажењу оптималног решења за проблем из неког специфичног домена дефинисаног промењивама. Када су промењиве дефинисане у домену целих бројева ради се о **комбинаторном оптимизационом** проблему. У случају када су промењиве дефинисане у скупу реалних бројева разликују се **оптимизациони проблем са ограничењима**, код којег је приликом проналажења оптималног решења потребно узети у обзир и додатне услове и/или ограничења, и

мултимодални оптимизациони проблем, код којег уместо једног оптималног решења постоји више локалних оптималних решења која су од интереса**.** Уколико је потребно одједном оптимизовати више проблема (фукнција) онда се ради о **вишекритеријумском оптимизационом проблему**. Решавање неких од ових проблема је доста лако и не захтева велики труд, док се други не решавају тако тривијално и њима се бави посебна дисциплина математике названа **операциона истраживања** [18].

Проблем учења настао је покушајем да рачунар у некој мери добије људску интелигенцију касније названу **вештачка интелигенција**. Проблеми учења се могу поделити на **учење са надзором** и **учење без надзора** у зависности од тога да ли за коришћење алгоритма потребно тренирање уз помоћ неког тестног узорка пре самог коришћења [18].

Слика 12- Пресек ВИ, ОИ и инжењерства представља ЕА

Пре приказа основног еволутивног алгоритма и представљања посебних типова ЕА важно је упознати се са основним појмовима који се јављају при раду са ЕА [19]:

- Скуп свих могућих решења у домену оригиналног проблема називају се **фенотипови**, а репрезентација појединачних решења у домену ЕА називају се **генотипови**. Постоји већи број синонима за елементе ова два скупа, али најчешће коришћени назив и за једне и за друге је **јединка. Репрезентација** представља мапирање из скупа фенотипова у скуп генотипова који представљају те фенотипове.
- Улога **функције евалуације** је постављање захтева којима је потребно да се јединка прилагоди, а најчешће је састављена као мера квалитета генотипа. Обично се назива **функција прилагођености** (енг. fitness function). Када се ради о проблему оптимизације функција евалуације се назива **функција циља** (енг.

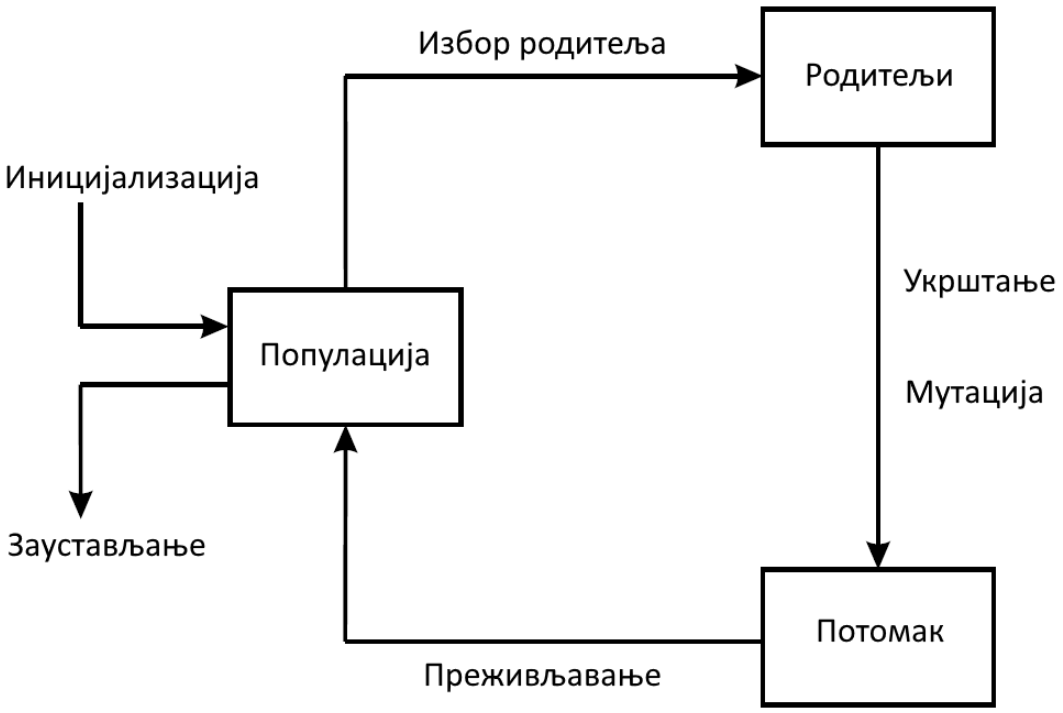
objective function) и представља управо функцију коју оптимизујемо или неку њену просту трансформацију.

- **Популација** чини мултискуп генотипова и служи за груписање могућих решења. Свака тренутна популација служи за прављење наредне генерације популације. Величина популације би требала да остане иста током животног века. **Разноврсност** популације представља меру која нам говори колико различитих решења постоји и та мера није јединствена. Најчешће је то број различитих фитнес вредности или број различитих генотипова, а некада се користе и статистичке мере попут ентропије.
- **Механизам избора родитеља** представља начин избора јединки заснован на њиховом квалитету како би се бољим јединкама омогућило да буду родитељи за следећу генерацију. Квалитетније јединке имају већу шансу да буду изабране за родитеље, али то није гарантовано услед стохастичког начина избора.
- Улога **оператора мењања** је креирање нових јединки од већ постојећих. Према арности разликујемо два оператора унутар ЕА
    - **Мутација** је унарни оператор мењања и представља операцију мењања која се примењује на генотип једне јединке и као резултат даје измењену јединку. Избор гена на коме ће се вршити измена је стохастички процес, а може се поновити на различитим деловима генотипа.
    - **Рекомбинација** или **укрштање** је бинарни оператор мењања који се примењује на две јединке тако што случајним избором саставња делове генотипа оба родитеља и тако креира једног или два нова потомка.
- Процес сличан процесу избора родитеља, али који се дешава после креирања потомака, назива се **механизам за преживљавање.** Овај процес је за разлику од механизма за избор родитења често детерминистички. Најлошије јединке из тренутне популације се мењају новим бољим јединкама које постају део следеће генерације популације.
- **Иницијализација** је процес формирања почетне генерације популације и дешава се случајним генерисањем задатог броја јединки. Како би се овај процес побољшао и генерисала „боља“ почетна популација могу се користити нека од хеуристика.
- Испуњавање **услова заустављања** може се десити на два начина
    - Долазак до унапред постављене фитнес вредности (или неке њене епсилон околине)
    - Када је неки од додатно постављених услова нарушен или прекорачен. Најчешћи додатни услови су максимално време извршавања, максималан број генерација и минималан раст или максималан пад фитнес вредности најквалитетније јединке.

Дакле, сви ЕА заснивају се на следећем процесу који се може преточити у општи алгоритам [20]:

1. На случајни начин се генерише почетна популација дефинисане (обично непоромењиве) величине,
2. Свака јединка из популације добија сопствену вредност прилагођености уз помоћ функције евалуације,

3. Следећи кораци се понављају све док нека јединка не задовољи постављени критеријум за вредности прилагођености или док није нарушен неки додатни услов
   3.1. Бирају се јединке са најбољим вредностима прилагођености и означавају се као родитељи
   3.2. Над родитељима се примењују оператор укрштања како би се добили потомци
   3.3. Над потомцима се примењује оператор мутације
   3.4. Врши се оцењивање новонасталих потомака додељивањем вредности прилагођености
   3.5. Кандидати са најлошијим вредностима прилагођености се избацују из популације, односно бивају замењени бољим



Слика 13 - Општи еволутивни алгоритам

Сви еволутивни алгоритми прате опште дефинисана правила, а разликују се у техничким детаљима који су прилагођени конкретној примени. Такође, већина ЕА заснивају се на случајној претрази простора. Неки од најчешће коришћених типова еволутивних алгоритама су:

- **Генетски алгоритми** (даље. ГА) као највећа класа ЕА представљају претраживачке хеуристике засноване на процесу природне селекције коришћењем оператора мутације, укрштања и селекције. ГА користе приближне фитнес вредности које се могу лако израчунати, па су погодни за решавање неких веома сложених проблема из реалног света. Мане генетских алгоритама су што не раде једнако добро на великим популацијама. Алгоритам је потребно прилаготити конкретном проблему како би извршавање било оптимално [20].
- **Еволуционе стратегије** су техника оптимизације заснована на прилагођавању и еволуцији, а за прилагођавање најчешће користе мутацију и селекцију. Заснивају

се на детерминистичком одабиру јединки само према рангирању вредности прилагођености, не узимајући у обзир конкретну вредност. Родитељи се замењују потомком само у случају да потомак има бољу вредност прилагођености, иначе се потомак одбацује. У једном циклусу мутације може се генерисати један или више потомака [18].

- **Генетско програмирање** представља технику представљања рачунарског програма скупом гена који се мењају и еволуирају коришћењем генетског алгоритма. Као резултат се добија рачунарски програм који је способан да изврши унапред дефинисани задатак. Сваки рачунарски програм је јединка у овом случају. Програми се традиционално представљају помоћу неке дрволике структуре или у новије време других репрезентација које се користе у линеарном генетском програмирању [20].
- **Еволуционо програмирање** налик је генетском програмирању, али у овом случају структура програма остаје непромењива, док се нумерички параметри прилагођавају. Најчешће коришћени оператор је мутација. Сваки родитељ генерише по једног потомка па се сваки пут селекција врши над двоструким бројем јединки него што садржи иницијална популација [20].

4.2. Примена еволутивних алгоритма у видео играма

Према дефиницији, сврха видео игре је да забави играче, али проблем се јавља услед тога што сваки појединачни играч поседује своју субјективну оцену тога шта је за њега забавно. У зависности од карактера, предходног искуства и/или вештина конкретног играча задаци и садржај видео игре морају бити постављени на другачији начин како би се повећали фактор забавности и неизвесности, а смањили тривијалности и једноличност.

Рачунарски вођени ликови (даље РВЛ) у жаргону названи „ботови“ (енг. bots) су један од најчешће коришћених начина да се игра учини забавнијом, а користе се како у играма за једног играча (енг. singleplayer), тако и у играма за више играча (енг. multiplayer). РВЛ представљају главну примену вештачке интелигенције у видео играма, а за циљ имају да опонашају праве играче који поседују различите нивое искуства и да се понашају што је могуће природније како би правим играчима били достојни противници или добри саборци.

Упоредо са увођењем РВЛ у видео игру потребно је прилагодити их конкретном играчу како би његова очекивања била задовољена и како би победа или испуњавање неког другог задатка у видео игри били одговарајуће тежине. За постизање максималног ефекта такође је потребно да сам садржај и правила видео игре такође буду прилагођена конкретном играчу. Овакав посао тешко је обављати ручно, а како то представља главну покретачку снагу која води ка побољшању дизајна и коначно ка бољим видео играма које ће очувати лојалност крајњих корисника, мора се наћи начин за то. Предходно наведено представља другу велику примену вештачке интелигенције у видео играма, а прикладан начин за имплементацију је примена еволутивних алгоритма.

4.2.1. Прилагођавање садржаја

Коришћење алгоритама и техника са могућношћу учења и прилагођавања као што су ЕА од кључне је важности за напредак индустрије видео игара.

Апликација	Генерисање	Мерење				Прилагођавање
		Садржај	Играч			
			Личност	Понашање	Искуство	
3Д Акција						
The Hunter	-	-	-	ДА	-	-
Rogue Troper	-	-	-	ДА	-	-
Аркадна						
Pac-Man	-	ДА	-	ДА	-	-
Super Mario	ДА	-	-	ДА	-	ДА
Платформер						
ANGELINA	ДА	ДА	-	-	-	-
Таблична						
LUDI	ДА	ДА	-	-	ДА	-
Shibumi	ДА	ДА	ДА	-	ДА	ДА
Слагалица						
Hour Maze	ДА	ДА	-	-	ДА	-
Отворени Свет						
Subversion	ДА	ДА	-	-	ДА	ДА

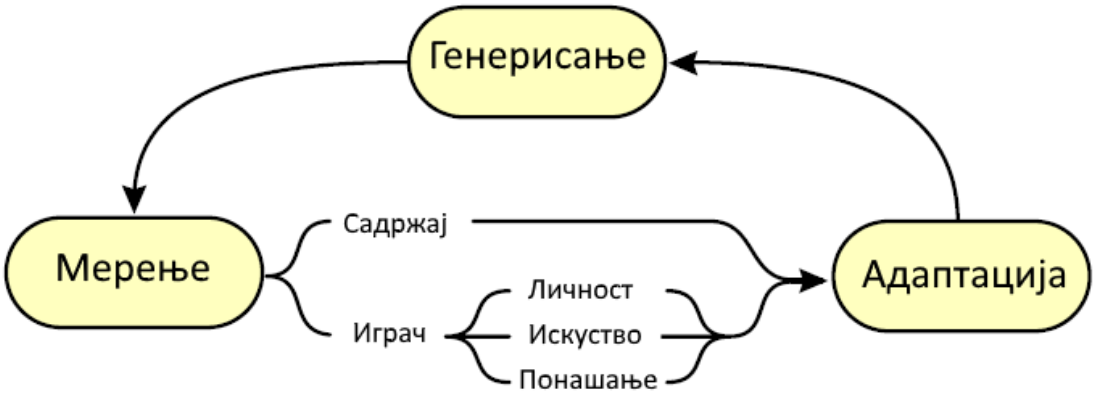
Слика 14 - Преглед коришћења ЕА у видео играма разичитог жанра

Зависно од типа садржаја и конкретног жанра видео игре за прилагођавање је потребно користити различите алгоритме који не дају увек једнаке резултате. Узроковано тиме квалитет прилагођавања видео игре за конкретног корисника је некада бољи, а некада лошији. Независно од избора, основни кораци за било који алгоритам су исти:

- генерисање новог садржаја и правила,
- мерење изгенерисаног садржаја и играча током адаптивног генерисања или као део дизајна система и оцењивања,
- прилагођавање са циљем побољшања корисничког искуства код играча.

4.2.1.1. Генерисање садржаја

Почетни корак у процесу генерисања садржаја у видео играма са могућношћу прилагођавања (даље ВИМП) је стварање новог садржаја и правила видео игре. Већина овог посла се може урадити потпуно аутоматизовано уз помоћ ПГС тако да рачунар преузима улогу креативног сарадника.



Слика 15 -Преглед процеса прилагођавања изгенерисаног садржаја

ПГС засновано на претрази (даље ПГС-ЗП), описано у поглављу 3.2, представља одговарајући механизам за прилагођавање садржаја видео игара „у лету“ (енг. on the fly) према захтевима играча. Систем поседује могућност учења и резултати ће бити бољи сваким наредним коришћењем. Задатак представља генерисање дела садржаја који може испунити одређене постављене циљеве специфичне за конкретног играча, а решење је управо део изгенерисаног садржај који може да задовољи задате циљеве. Итеративни процес генерисања се наставља све док се не дође до решења или док неки други критеријум заустављања не буде испуњен.

Приликом коришћења функција прилагођености и репрезентација решења које су релативно једноставне ЕА показују јако добре карактеристике. За проблеме који имају сложенија решења и код којих је потребно извршавати сложеније провере приликом рачунања фитнес вредности тешко је дизајнирати добар ЕА и уобичајено тај процес траје нешто дуже. Такође сама стохастичка природа генерисања и мутације унутар ЕА може изазвати да се чак и код најбоље дизајнираних решења не добијају увек најбољи резултат, па је то разлог све већег коришћења ЕА у развојној фази, када се има времена за откривање недостатака и додатна подешавања.

Посебан тип ЕА помоћу којег већи проблеми могу бити решени разбијањем на мање проблеме који се онда посебно решавају назива се **кооперативна коеволуција** (енг. Co-operative co-evolution, даље КОКЕ) [5]. Сваки подпроблем се гледа као потпуно независан еволутивни проблем са сопственом популацијом и функцијом евалуације. Како би се оценио квалитет решења неког подпроблема најбоље јединке из свих осталих подпроблема спајају се у решење оригиналног проблема и помоћу функције евалуације процењује квалитет целокупног решења и као и квалитет сарадње тог решења са осталим решењима подпроблема. КОКЕ је веома примењив у видео играма које се природно могу поделити на више подпроблема или подсистема који се могу дизајнирати одвојено, а онда спојити у једно велико свеобухватно решење.

4.2.1.2. Мерење и евалуација

Наредни корак у процесу прилагођавања је мерење генерисаног садржаја, што укључује:

- мерење квалитета генерисаног садржаја видео игре и правила игре према унапред одређеним критеријумима или задатим циљевима,
- мерење начина на који играч игра игру уз мерење спољашњих фактора.

У већини случајева уобичајено се користи један приступ, или мерење садржаја или мерење активности играча, али оба приступа могу подједнако допринети при дизајнирању видео игре. Мерење може играти три различите улоге у генерисању адаптивног система видео игре:

- **мерење прилагођавања** – систем врши мерења генерисаног садржаја и циљног играча, како би садржај прилагођен за тог играча био испоручен,
- **формативно мерење** – дизајнери система тестирају квалитет генерисаног садржаја и реакцију играча на садржај и/или прилагођавање садржаја што може бити од помоћи за генерисање новог садржаја,
- **збирно мерење** – радни систем се оцењује у смислу квалитета генерисаног садржаја и реакција играча на садржај и/или прилагођавање садржаја.

Мерење прилагођавања је аутономно, спроведено уз помоћ система за дизајн игре, док се формативно и збирно мерење спроводе од стране људи и код таквих врста мерења могуће је доћи до информација које нису доступне при аутономном мерењу. Приликом играчевог субјективног описа онога што је доживео током играња видео игре разумевање доживљаја је доста квалитетније и тачније, али су тако добијене информације тешко прилагодљиве за обраду на рачунару. Аутономна мерења унутар саме игре и физиолошка мерења спроведена помоћу уређаја су доста прецизнија и лакше се врши њихова каснија обрада.

Квалитет видео игре је веома важан како би заинтересовао играче и задржао их, али испитивање квалитета генерисане игре може бити веома тешко, јер је зависно од контекста и мења се у зависности од играча. Један од приступа може бити дефинисање квалитетих метрика које ће помоћи не само при претрази током генерисања садржаја већ и приликом пројектовања и евалуације. Често коришћена алтернативна техника је „плејтестинг“ (енг. playtesting) где се информације прикупљају од играча који спроводе тестирање.

Четири кључне карактеристике апстрактних игара су [5]:

- **дубина** (енг. depth) - капацитет игре да омогући играње на различитим нивоима вештина и обезбеђивања „награда“ за непрекидно напредовање,
- **јасноћа** (енг. clarity) - лакоћа са којом играч може разумети правила и планирати потезе,
- **драма** (енг. drama) - могућност повратка играча са зачеља како би евентуално однео победу,
- **одлучност** (енг. decisiveness) – лакоћа којом се игра може завршити када је познат победник.

Још неке корисне мере укључују **несигурност**, **равнотежу** и **дужину игре**.

Када се ради о подацима о играчу они могу бити прикупљени пре почетка игре, како би било могуће генерисати одговарајући садржај за нову игру или током игре како би надолазећи садржај у тренутној игри био био прилагођен. Податке је најлакше скупити

у **дневнике игре** (енг. game logs), који могу садржати стања и догађаје настале током игре или податке о играчу. Дневници игре се могу користити појединачно, а могу се користити и за агрегацију што је значајно за мерење код игара у којима учествује више играча.

Уместо да прикупљени подаци о играчу буду прослеђени генератору садржаја они се преводе у апстрактни **модел играча** који представљају податке о играчу више прилагођене следећем генерисању садржаја. Овакав модел садржи не велики број података о навикама корисника и категоризације преведене на једноставнији запис како би касније могле бити лако коришћене као улазни подаци за ВИ. Разликујемо три општа типа модела играча [5]:

- **Модел личности** (енг. personality model) описује играча у смислу индивидуалних психолошких карактеристика и описа саме личности играча. Подаци о играчу се помоћу неке од техника машинског учења повезују са његовом личности. Овај модел може бити коришћен независно од видео игре током које је настао.
- **Модел искуства** (енг. experience model) служи за описивање онога што корисник доживљава током неког периода играња. Овај модел је везан за конкретну видео игру у којој настаје и не може бити искоришћен у другим видео играма. Као и код модела личности, уз помоћ машинског учења, овај модел може послужити како би се побољшало знање о повезаности акција корисника и корисничког искуства.
- **Модел понашања** (енг. behavioural model) показује шта је играч учинио, како у игри тако и друштвено или физички. Овакав тип модела је за разлику од предходна два модела доста ближи акцијама корисника и може бити направљен из играчких логова потпуно аутономно.

Током фазе дизајнирања и оцењивања прилагодљивог система игре мерења могу бити коришћена и у сврху **мерење прилагођавања.** Узимање у обзир искуства играча након промене садржаја видео игре може показати да ли је та промена позитивно или негативно утицала на играча.

4.2.1.3. Прилагођавање

Након што су извршена мерења и успешно направљени модели играча прелази се на последњи корак у процесу прилагођавања игре, а то је испоручивање прилагођеног садржаја играчу. Како би одабрани садржај био што прикладнији приликом процеса испоручивања прилагођеног садржаја користи се већ генерисан садржај, мерења квалитета садржаја и процена циљног играча. У појединим системима процес прилагођавања може бити заустављен одмах након испоруке прилагођеног садржаја, док у осталим системима то може бити непрекидан процес у коме се надолазећи садржај прилагођава на основу сталних мерења реакција корисника на предходно прилагођени садржај.

Приликом коришћења техника ПГС-ЗП основно што је потребно је да функција или алгоритам претраге буде добро дефинисан. Мада је често довољно коришћење једноставних еволутивних алгоритама, неретко се коришћењем мало компликованијих и алгоритама прилагођенијих конкретном садржају добијају знатно бољи резултати.

Осим функције претраге, кључно је дефинисати функцију прилагођености и начин репрезентације садржаја.

Функција прилагођености, према којој се врши прилагођавање генерисаног садржаја, у овом случају креира се на основу модела играча, а могу се разликовати три кључне класе функција прилагођености [21]:

- директне функције прилагођености (даље ДФП),
- функције прилагођености засноване на симулацији (даље ФПЗС),
- интерактивне функције прилагођености (даље ИФП).

Код употребе ДФП из генерисаног садржаја се узимају неке од карактеристика које се повезују са вредношћу прилагођености тог садржаја. Повезивање може бити директно или индиректно, али се свакакако не врше компликована израчунавања.

Како није увек очигледно како направити ДФП, коришћењем интелигентног агента који симулира играње видео игре може бити креирана ФПЗС. Интелигентни агент може бити статичан или се може мењати током играња прилагођавајући се, а од брзине и квалитета прилагођавања зависи и квалитет настале функције прилагођености тестираног садржаја. Узимајући у обзир предходно наведено, очигледно је да су ФПЗС повезане са моделом играча насталог током играња видео игре.

ИФП се заснивају на интеракцији садржаја са играчем и вредност прилагођености садржај добија током играња. Информације о интеракцији могу бити експлицитно прикупљене од играча или могу бити извршена имплицитна мерења. Као и код модела играча, информације прикупљене од играча су доста квалитетније, али је рачунарска обрада информација насталих помоћу имплицитних мерења много лакша.

Избор јединке која има максималну вредност функције прилагођености врши се помоћу ЕА уобичајеном претрагом простора уз коришћење предходно дефинисане функције претраге, а како би процес претраге био што је могуће оптималнија потребно је одабрати одговарајући начин за репрезентацију јединки.

Директно кодирање (енг. direct encodings) је тип репрезентације код које је дужина генотипа линеарно пропорционална дужини фенотипа и сваки посебан део генотипа је повезан са тачно одређеним делом фенотипа.

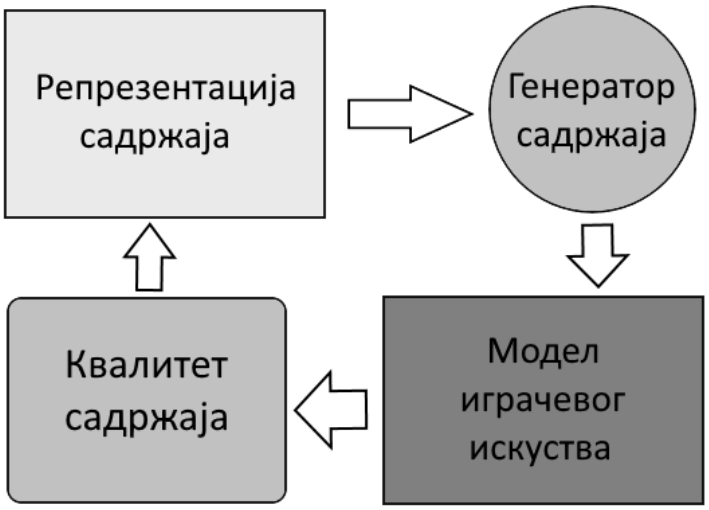
Насупрот директном кодирању, **индиректно кодирање** (енг. indirect encodings) је тип репрезентације код које генотип и фенотип не морају бити пропорцијални услед тога што повезивање није линеарно. Који год тип репрезентације био коришћен важно је одабрати прикладну величину саме репрезентације, која не треба да буде ни предуга, а ни прекратка. Такође важна карактеристика репрезентације је висок **локалитет**, што означава својство да мале промене на генотипу обично треба да узрокују и мале промене на самој репрезентацији и вредности прилагођености.

4.2.1.4. Технике за прилагођавање садржаја

Методе засноване на решавању (енг. Solver-based methods) такође су засновани на идеји претраге простора приликом генерисања садржаја који задовољава постављена

ограничења, али је притом претрага вођена решавачима ограничења. Ограничења могу бити постављена математички или логички, а решавачи осим што могу бити засновани на еволуционим израчунавањима, могу бити засновани на САТ решавачима. Главна разлика у односу на класичне алгоритме засноване на претрази је то што се уместо претраге целог простора решења врши само делимична претрага и током процеса се врши одсецање решења све док не остану само решења која задовољавају све постављене услове.

ПГС засновано на искуству (енг. experience-driven PCG, даље ПГС-ЗИ) представља унапређени приступ за генерисање садржаја повезујући корисничко искуство са генерисаним садржајем. Подаци о корисничкм искуству прикупљени су неком од техника



Слика 16 - ПГС засновано на искуству

Дизајнирање механизма који је у могућности да се адаптра може бити третиран и као проблем машинског учења, где је потребно да систем стекне знање да податке које добија од стране играча повеже са садржајем игре који задовољава задате критеријуме квалитета за конкретну групу којој играч припада. Уобичајено је да типови улазних и излазних података буду категорички или скаларни, али уопштено то могу бити и много компликованије структуре података које је пре обраде потребно прилагодити механизмима за машинско учење, а управо модели играча решавају тај проблем.

5. Пример имплементације видео игре

Видео игра имплементирана за потребе овог мастер рада носи радни назив „Крадљивац воћа“ (енг. Fruit Stealer) и припада категорији платформских видео игара. Замишљена је као игра са неограниченим кретањем у којој се карактер којим управља играч креће унапред унутар тродимензионалног простора са циљем да сакупи што више воћа и самим тим и више поена, а да притом мора избегавати препреке и задржати се на подлози којом се креће. Генерисање подлоге, распореда и типа објеката вршено је коришћењем процедуралног начина генерисања садржаја.

Слика 17 - Приказ изгледа видео игре

5.1. Коришћене технологије

Приликом израде видео игре коришћено је развојно окружење Јунити (енг. Unity). Коришћена је верзија 2018.2 овог развојног окружења за Виндоуз (енг. Windows) оперативни систем.

Скрипте коришћене у видео игри писане су у C# програмском језику, а као едитор је коришћено Вижуал Студио 2018 (енг. Visual Studio) развојно окружење.

За верзионисање кода коришћен је ГитХаб (енг. GitHub) и игра је расположива за скидање као софтвер отвореног кода на адреси http://github.com/JupikeA/FruitStealer.

5.2. Основна верзија видео игре

Генерисање садржаја у основној верзији видео игре вршено је уживо током играња видео игре коришћењем генератора псеудо-случајних бројева и садржај није накнадно

прилагођаван конкретном играчу. Алгоритми који су коришћени поседују велики степен контроле и флексибилности. У зависности од подешавања садржај може бити генерисан детерминистички или стохастички. Детерминистичко генерисање садржаја значајно је приликом тестирања видео игре, јер за исте параметре даје идентичан садржај, а како би то било постигнуто за добијање случајно генерисаних бројева коришћена је помоћна статичка класа ***MathHelper*** која има могућност конфигурисања почетног параметра.

```
public const bool DeterministigPCG = false;

public const int DeterministicSeedNumber = 5;

public static int RandomSeedNumber = !DeterministigPCG ? (int)DateTime.Now.Ticks :
5;
```

Исечак кода 1 - Конфигурисање начина генерисања садржаја

Коришћењем стохастичког начина генерисања садржаја може се добити велики број различитих нивоа, а увођењем мерења садржаја и памћењем само једног броја који представља јединствени идентификатор нивоа могуће је вршити успешну претрагу простора како би се ново-генерисани садржај што више личио ономе у нивоима са високим оценама.

Управљање генерисањем садржаја врше генератори садржаја за сваку групу повезаних објекта. У овом случају сав садржај везан је за подлогу, па је генерисан помоћу ***TerrainGenerator*** изведену из базне класе ***MovableGenerator***.

```
public abstract class MovableGenerator : IGenerator<MovableBase>

{

    protected Vector3 position;

    public abstract Queue<MovableBase> Initialize();

    public abstract MovableBase GenerateNext();

}
```

Исечак кода 2 - Приказ базне класе за генерисање садржаја

Садржај који је овом приликом генерисан може бити подељен на обавезни и необавезни. Обавезни садржај чине подлога, воће и замке. Необавезни садржај чине објекти који су украсне природе, а не утичу на сам ток игре пошто карактер приликом интеракције са

њима не може добити ни негативне ни позитивне поене, односно не може бити уништен што је случај приликом додиривања неке од замки или приликом испадања са подлоге.

Додатни садржај може лако бити додат у неку од предефиснисаних категорија само додавањем модела тог садржаја. Нова категорија садржаја такође може бити додата независно од предходно додатог садржаја.

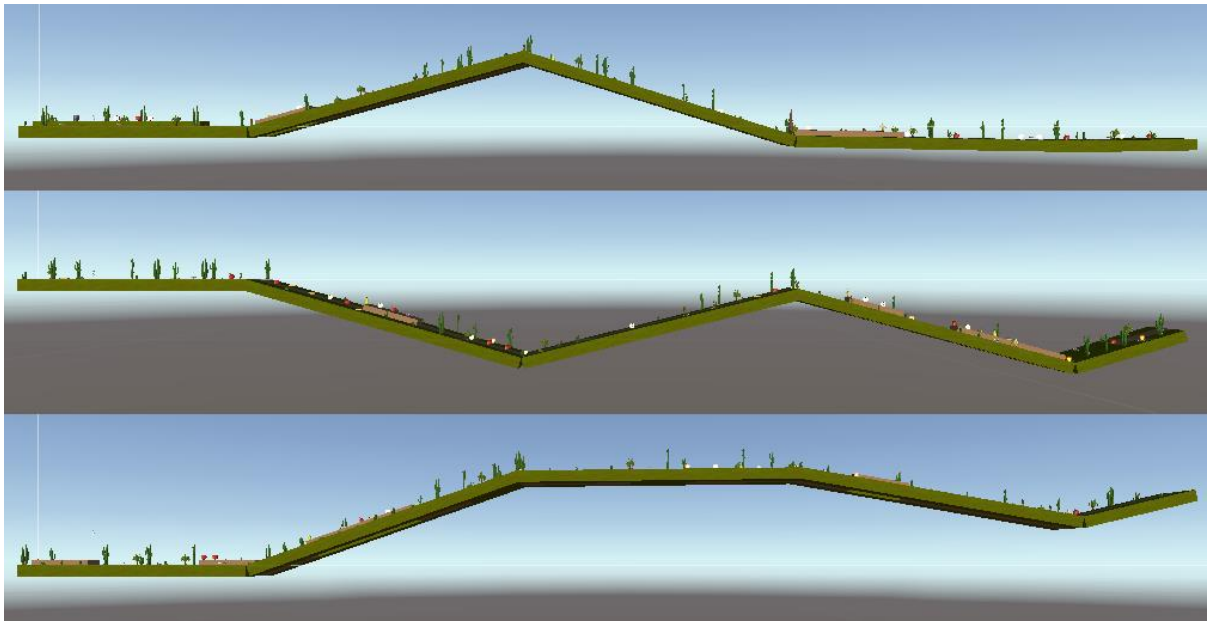
5.2.1. Подлога

Приликом генерисања подлоге по којој се карактер креће, пажња првенствено мора бити усмерена на то да се карактеру омогући непрекидно кретање. Подлога не сме садржати превелике размаке између делова, уколико се састоји из више делова, а сваки посебни део мора бити конструисан тако да омогући пролаз карактера од почетка до краја тог дела. Подлога служи и као водиља за генерисање осталог садржаја па је потребно да приликом генерисања подлоге у обзир буде узет и садржај који ће бити изгенерисан.

У основној верзији видео игре генерисана подлога је у потпуности повезана и карактер се може кретати непрекидно унапред. Генерисање подлоге могуће је параметарски подешавати по следећим параметрима:

- дужина,
- ширина,
- висина,
- број трака,
- број сегмената са константним нагибом,
- минималан и максималан нагиб терена,
- тип подлоге.

Првих пет параметра не мењају се током игре. Нагиб подлоге се мења на унапред одређени број сегмената, а избор угла врши се случајним избором између унапред дефинисаног минималног и максималног нагиба. Тип подлоге се такође бира случајним избором, али са предефинисаном вероватноћом. Избор подлоге, иако то није случај у основној верзији видео игре, може утицати на кретање карактера и бити или додатна препрека или додатна помоћ играчу.



Слика 18 - Примери профила дела генерисаног терена

5.2.2. Објекти

Остали објекти генерисани у основној верзији видео игре, као што је већ речено у предходном потпоглављу, подељени су на обавезне и необавезне. Воће (класа ***Fruit***) је једини садржај који се сакупља и тиме доноси позитивне поене. Обавезни садржај чине још и замке (класа ***Trap***) које при контакту уништавају карактер или му доносе негативне поене. Како би се обезбедила проходност однос и распоред воћа и замки такође је генерисан случајним избором са унапред одређеном вероватноћом.

Необавезан садржај чине вегетација (класа ***Vegetation***) и тврди објекти (класа ***HardObject***) који у одређеној мери спречавају карактер да се креће унапред, али никако нису незаобилазни, већ су више ту да игру учине занимљивијом и визуелно допадљивијом.

Свака од предходно наведених категорија садржаја генерисана је потпуно случајним избором неког од унапред учитаних модела. Уколико случајни избор није довољно добар, могуће је искористити случајни избор са одређеном вероватноћом за сваки објекат. За додавање нових објеката у одређену категорију довољно је додати жељени модел у директоријум те категорије.

Додатно је могуће „налепити“ скрипте на сваки од објеката како би одређене акције биле извршене при контакту са тим објектима, као што је случај са замкама које експлодирају након што дођу у контакт са карактером и тако се завршава игра. Воће, са друге стране садржи скрипте које омогућавају подешавање поена који ће бити сакупљени приликом сакупљања тог воћа.

5.2.3. Запажања и закључци

Основну верзију видео игре тестирала су два играча различитог нивоа искуства. Приликом тестирања видео игре вршена су субјективна мерења која су узета од оба играча која су вршила тестирање. Први играч је поседовао средњи ниво искуства, док други играч није поседовао искуство у оваквом типу видео игре. Након анализе извршених субјективних мерења добијених усменим путем од играча закључено је следеће:

- видео игра је занимљива и привлачна и играчи би наставили да је играју,
- видео игра је тешка чак и на самом почетку,
- објекти у видео су у неким случајевима изгенерисани превише шаблонски (пар истих објеката су изгенерисани један за другим, па их нема дуго времена).

Упоредо са субјекивним мерењима извршена су и аутоматска мерења која су имала за циљ следеће:

- мерење броја изгенерисаних објеката по типу објекта у времену од 60 секунди,
- мерења времена проведеног у видео игри пре наиласка на замку,
- мерење броја прикупљених поена,
- мерење броја прикупљених објеката.

Након анализе аутоматски сакупљених података добијени су следећи резултати:

- Први играч провео је просечно 18.82 секунди играјући основну верзију видео игре пре него што је налетео на неку од замки, том приликом је просечно сакупио 97 поена односно 12 објеката. Просечан број сакупљених поена по секунди је 5.
- Други играч провео је просечно 21.02 секунди пре наиласка на замку, а при том је просечно сакупио 76 поена односно 10 објеката. Просечан број сакупљених поена по секунди је 4.

Анализа података о броју генерисаних објеката показала је да су објекти генерисани равномерно и да не постоје нека претерана одступања од очекивања. Само за један од објеката је једном забележено да је генерисан чак 20% више од очекиваног у односу на остале објекте тог типа.

Након узимања у обзир свих предходно наведених анализа и података закључено је следеће:

- потребно је да видео игра на почетку буде лакша, а да са временом постаје све тежа,
- приликом промене фактора тежине видео игре у обзир се могу узети и аутоматски сакупљени подаци о играчу за предходни период играња или за неколико предходних играња,
- потребно је поправити начин генерисања објеката тако да се што мање примећују шаблони узроковани природом ПСГБ који су коришћени приликом генерисања,
- потребно је ублажити прелазе између нагиба приликом генерисања подлоге.

У напредној верзији видео игре биће покушано исправити недостатке основне верзије видео игре.

5.3. Напредна верзија видео игре

Први корак за прилагођавање видео игре је прикупљање и праћење неких од података који су доступни током играња. Помоћу скрипте ***StatisticsManager*** прикупљени су следећи подаци:

- време протекло од почетка играња,
- укупан број сакупљених објеката,
- укупан број сакупљених поена,
- просечан и максималан број сакупљених објеката по секунди,
- просечан и максималан број сакупљених поена по секунди.

Прикупљени подаци о играчу, као и подаци о генерисању објеката, уписују се у дневник-датотеку како би могли да се употребе и након завршетка игре, али циљ је да првенствено буду искоришћени током игре.

5.3.1. Подлога

Како су прелази између нагиба подлоге били превише оштри и неприродни, употребљен је прост еволутивни алгоритам који је побољшао генерисање подлоге. Постављени циљ био је ублажити неприродни изглед подлоге и омогућити бољу контролу генерисања.

За решавање проблема оштрих прелаза као основни алгоритам за генерисање подлоге употребљена је синусна функција.

```
private float GetCurrentAngle()

{

    float currentX = (nextChange - generatedItems) * Configuration.RoadwayLength;

    double currentSinus = Math.Sin(currentX * Math.PI /
       (Configuration.RoadwayLength * 12.5 / currentParametars.CorrectionX));

    return (float)(Math.Asin(currentSinus) * ToDegreeFactor /
       (currentParametars.CorrectionY * 2.0f));

}
```

Исечак кода 3 - Синусна функција за генерисање углова

Контрола синусне функције вршена је коришћењем параметара ***CurrentX*** и ***CurrentY*** из класе ***TerrainParametars***. Промена ових параметара вршена је када тренутна вредност ***X*** достигне неки умножак од ***π/2*** како би се и на тим местима добили глатки прелази. Избор нових параметара вршен је управо еволутивним алгоритмом.

Еволуивни алгоритам се састоји из три дела:

1. генерисање нове популације

2. оцена генерисаних параметара
3. случајан избор између најбоље оцењених параметара

Генерисање нове популације параметара вршено је помоћу оператора укрштања и оператора мутације. Избор најбољих родитеља од који ће бити коришћени за укрштање вршен је при сваком новом генерисању параметара. Родитељ са најлошијом оценом из скупа најбољих родитеља замењен је са најбољим параметром из тренутне генерације уколико је тај параметар боље оцењен.

Приликом укрштања случајним избором су из скупа најбљих родитеља бирана два родитеља која ће бити укрштена. Исечак кода 4 приказује да је могуће да дете у једном случају наследи оба параметра једног од родитеља или да не наследи ни један параметар од родитеља у супротном случају. Наслеђивање параметара се врши случајним избром са вероватноћом тако да се променом те вероватноће мења и наследни фактор.

```
var firstParent = bestParentList[MathHelper.Range(0, bestParentList.Count - 1)];

var secondParent = bestParentList[MathHelper.Range(0, bestParentList.Count - 1)];

var correctionX = MathHelper.RandomBool(0.5) ? MathHelper.Range(1.0f, 10.0f) :

(MathHelper.RandomBool() ? firstParent.CorrectionX : secondParent.CorrectionX);

var correctionY = MathHelper.RandomBool(0.5) ? MathHelper.Range(1.0f, 5.0f) :
(MathHelper.RandomBool() ? secondParent.CorrectionY : firstParent.CorrectionY);


child = new TerrainParametars(correctionX, correctionY);
```

Исечак кода 4 - Оператор укрштања

Након генерисања нове генерације коришћен је оператор мутације који је на одређеном делу генерације мењао параметре ***CurrentX*** и ***CurrentY***. Као и оператор укрштања и оператор мутације поседује подешавање вероватноће да мутација буде извршена, али је та вероватноћа доста мања (20% у овом примеру) па се може десити да ни једна мутација не буде извршена.

```
for (int i = 0; i < this.populationSize; i++)

    if (MathHelper.RandomBool(0.2f))

    {

        var child = newGeneration[i];

        newGeneration[i] = new TerrainParametars(child.CorrectionY * 2.0f,
                                    child.CorrectionX / 2.0f);

    }
```

Исечак кода 5 - Оператор мутације

Након генерисања нове популације врши се оцењивање параметара те популације коришћењем функција прилагођености приказане на Исечак кода 6. Функција је

конструисана тако да најбољу оцену добијају сегменти велике дужине и мале висине, а најлошију оцену кратки сегменти велике висине.

```
float mark = 1.0f / child.CorrectionX * 2 - 1.0f / child.CorrectionY * 1.5f;
```

Исечак кода 6 - Функција прилагођености

Коначни избор новог параметра вршен је случајним избором неког од параметара са најбољим оценама. Избор је вршен из скупа величине не већем од 25% од величине читаве нове популације, што је у овом случају 5.

```
List<int> bestCandidateIndex = newGenerationMarks.OrderByDescending(item =>
item.Value).Take(this.populationSize / 4).Select(item => item.Key).ToList();

int winnerIndex = bestCandidateIndex[MathHelper.Range(0,bestCandidateIndex.Count -
1)];

TerrainParametars winner = newGeneration[winnerIndex];
```

Исечак кода 7 - Избор коначног решења

Приликом генерисања сваког посебног угла вршене су следеће корекције:

1. Локална мутација случајним повећавањем или смањивањем угла која не утиче на оцењивање параметара задатог сегмента, а служи како би подлога добила случајне избочине и неправилности као у реалности.
2. Ограничење углова тако да не буду већи од задатог максималног угла или мањи од задатог минималног угла.

Додатне корекције могу се вршити фактором ***RoadwaySlopesFactor*** који је могуће подешавати тако да се добије "брдовитија" или равнија подлога.

Као помоћ при одабиру добрих параметара за генерисање подлоге коришћена је минимална и максимална надморска висина из дневник-датотеке (координата Y вектора који одговара позицији подлоге).

Еволутивни алгоритам је, као што је и очекивано показао добре резултате и након неколико генерација просечна оцена нових параметара је порасла. Висинске разлике и непредвиђена „дивљања" углова која су проузроковала превише узбрдица или низбрдица су смањена, а опсег и интензитет нагиба се могу успешно контролисати уз помоћ параметара.

5.3.2. Објекти

Непотребна почетна тежина игре уклоњена је увођењем фактора ***TrapGenerationFactor*** који коригује вероватноћу генерисања замки у односу на остале објекте, а повећава се временом проведеним у видео игри. Циљ је био да се играчу омогући да на почетку прикупи што више објеката који ће му донети поене. Такође је направљена измена при

генерисању замки тако да је на почетку генерисан мањи број замки веће величине, а како време напредује генерисан је већи број мањих замки. Контролисање величине замки вршено је помоћу фактора ***TrapScaleFactor***.

Како се број воћа смањивао током времена тако је и број поена које оно доноси повећаван, али је такође и повећавана брзина кретања карактера.

Шаблонско генерисање воћа и осталих објеката које је било примећено у основној верзији видео игре превазиђено је једноставним постављањем услова да нови генерисани објекат не може бити неки од предходно изгенерисаних објеката тог типа. Тиме је уједно и још више уравнотежен однос броја изгенерисаних објеката истог типа што се може видети у прилогу .

5.3.3. Закључак

Предходне измене позитивно су утицале на оба играча која су тестирала основну верзију видео игре, па су приликом тестирања напредне верзије видео игре постигнути следећи резултати:

- Први играч просечно је провео 46.69 секунди играјући наппредну верзију видео игре и том приликом је просечно сакупио 582 поена односно 55 објеката. Просечан број сакупљених поена по секунди је 11.
- Други играч провео је просечно 45.61 секунди пре наиласка на замку у напредној верзији видео игре, а при том је просечно сакупио 501 поена односно 53 објекат. Просечан број сакупљених поена по секунди је 10.

За постизање максималног прилагођавања и повећања задовољства код играча потребно је узети у обзир више параметара и извршити више тестирања на широкој популацији, али очигледно је да чак и најмања прилагођавања имају позитиван ефекат.

6. Закључак

У овом раду представљене су технике за процедурално генерисање садржаја код видео игара и показане су њихове предности и мане. Такође су приказани и начини како да се неке од наведених мана уклоне и представљене технике побољшају и прилагоде конкретној ситуацији.

Као што је и показано приликом прављења основне верзије видео игре, технике за процедурално генерисање садржаја не могу дати савршене резултате без накнадног прилагођавања генерисаног садржаја. Прилагођавање превасходно може послужити за исправљање неправилности насталих приликом генерисања садржаја. Финија подешавања везана за конкретног играча изискују и додатна улагања за прикупљање и обраду података од значаја, али у великој мери могу побољшати кориснички доживљај. Стога је потребно наћи компромис, тј. оптималан однос уложеног и добијеног. Употреба еволутивних алгоритама такође може донети додатна побољшања и повећати контролу приликом генерисања садржаја.

Ипак, општи закључак је да приликом прављења видео игара треба користити много озбиљније технике и сложеније еволутивне алгоритаме за процедурално генерисање садржаја како би резултати били прихватљиви. Да би се постигла максимална ефикасност, потребно је и добро познавање таквих техника и њихових предности и мана уз неопходнни тестни период у коме се могу утврдити потенцијалне неправилности.

Накнадна истраживања на ову тему могу увелико помоћи при коришћењу и унапређењу постојећих или развоју нових техника за процедурално генерисање садржаја. У том смислу, видео игра имплементирана приликом рада на овој мастер тези, доступна на адреси http://github.com/JupikeA/FruitStealer, представља добру основу за даља истраживања и анализу процедуралног генерисања садржаја у видео играма.

7. Прилози

7.1. Прилог 1 – Статистика генерисаних објеката у основној верзији видео игре

Тип	**Име**	**Број**	**% од укупног**	**Просечан %**	**Разлика**
Fruit	Apple	21	9.5%	11.1%	1.6%
Fruit	AppleHalf	23	10.5%	11.1%	0.7%
Fruit	Banana	31	14.1%	11.1%	3.0%
Fruit	BananaOpen	21	9.5%	11.1%	1.6%
Fruit	Kiwi	28	12.7%	11.1%	1.6%
Fruit	KiwiHalf	27	12.3%	11.1%	1.2%
Fruit	Pear	26	11.8%	11.1%	0.7%
Fruit	Strawberry	23	10.5%	11.1%	0.7%
Fruit	StrawberryHalf	20	9.1%	11.1%	2.0%
HardObject	Chopped_Wood_Pile	1	14.3%	20.0%	5.7%
HardObject	Rock_Small_01	1	14.3%	20.0%	5.7%
HardObject	Rock_Small_02	1	14.3%	20.0%	5.7%
HardObject	Wooden_Barrel (1)	1	14.3%	20.0%	5.7%
HardObject	Wooden_Crate	3	42.9%	20.0%	22.9%
Trap	Dynamite	27	15.5%	20.0%	4.5%
Trap	Pf_Trap_Cutter	36	20.7%	20.0%	0.7%
Trap	Pf_Trap_Fire	44	25.3%	20.0%	5.3%
Trap	Pf_Trap_Needle	38	21.8%	20.0%	1.8%
Trap	SteamSpray	29	16.7%	20.0%	3.3%
Vegetation	Cactus_Leafy_01	44	11.9%	8.3%	3.6%
Vegetation	Cactus_Leafy_02	25	6.8%	8.3%	1.6%
Vegetation	Cactus_Leafy_03	23	6.2%	8.3%	2.1%
Vegetation	Cactus_Leafy_04	30	8.1%	8.3%	0.2%
Vegetation	Cactus_Leafy_05	33	8.9%	8.3%	0.6%
Vegetation	Cactus_Short_01	30	8.1%	8.3%	0.2%
Vegetation	Cactus_Short_02	24	6.5%	8.3%	1.8%
Vegetation	Cactus_Short_03	36	9.8%	8.3%	1.4%
Vegetation	Cactus_Tall_01	32	8.7%	8.3%	0.3%
Vegetation	Cactus_Tall_02	28	7.6%	8.3%	0.7%
Vegetation	Cactus_Tall_03	38	10.3%	8.3%	2.0%
Vegetation	Cactus_Tall_04	26	7.0%	8.3%	1.3%
					2.9%

7.2. Прилог 2 – Подаци о играчима током играња основне верзије видео игре

Играч 1 - Средњи ниво искуства			
Време (секунди)	**Поена сакупљено**	**Обеката сакупљено**	**Поена по секунди**
4.76	18	2	3.78
19.57	74	11	3.78
44.82	197	25	4.40
4.16	13	2	3.13
1.41	0	0	0.00
9.54	74	8	7.76
17.67	98	11	5.55
28.94	77	9	2.66
10.29	45	5	4.37
8.39	78	9	9.30
51.53	270	29	5.24
5.52	65	8	11.77
21.80	97	12	4.45
12.18	59	8	4.84
25.97	69	9	2.66
25.71	103	10	4.01
5.24	47	5	8.97
31.37	201	23	6.41
37.10	233	30	6.28
10.42	124	14	11.90
18.82	**97**	**12**	**5.56**

Играч 2- Почетни ниво искуства			
Време (секунди)	**Поена сакупљено**	**Обеката сакупљено**	**Поена по секунди**
19.53	93	14	4.76
16.27	19	2	1.17
5.18	0	0	0.00
54.39	107	15	1.97
12.27	0	0	0.00
21.17	35	3	1.65
35.21	84	12	2.39
21.90	80	9	3.65
36.91	118	17	3.20
5.40	29	3	5.37

26.04	191	23	7.34
12.85	11	1	0.86
24.77	223	27	9.00
1.43	0	0	0.00
2.60	28	4	10.77
4.69	58	12	12.36
49.05	174	23	3.55
20.16	64	8	3.18
45.99	192	29	4.17
4.63	12	1	2.59
21.02	**76**	**10**	**3.90**

7.3. Прилог 3 – Сатистика генерисаних објеката у напредној верзији видео игре

Тип	**Име**	**Број**	**% од укупног**	**Просечан %**	**Разлика**
Fruit	BananaOpen	36	11.39%	11.11%	0.28%
Fruit	Kiwi	35	11.08%	11.11%	0.04%
Fruit	Apple	35	11.08%	11.11%	0.04%
Fruit	Banana	35	11.08%	11.11%	0.04%
Fruit	KiwiHalf	35	11.08%	11.11%	0.04%
Fruit	Pear	35	11.08%	11.11%	0.04%
Fruit	Strawberry	35	11.08%	11.11%	0.04%
Fruit	StrawberryHalf	35	11.08%	11.11%	0.04%
Fruit	AppleHalf	35	11.08%	11.11%	0.04%
Vegetation	Cactus_Leafy_02	31	8.40%	8.33%	0.07%
Vegetation	Cactus_Leafy_03	31	8.40%	8.33%	0.07%
Vegetation	Cactus_Leafy_04	31	8.40%	8.33%	0.07%
Vegetation	Cactus_Tall_03	31	8.40%	8.33%	0.07%
Vegetation	Cactus_Leafy_05	31	8.40%	8.33%	0.07%
Vegetation	Cactus_Short_01	31	8.40%	8.33%	0.07%
Vegetation	Cactus_Short_02	31	8.40%	8.33%	0.07%
Vegetation	Cactus_Short_03	31	8.40%	8.33%	0.07%
Vegetation	Cactus_Tall_01	31	8.40%	8.33%	0.07%
Vegetation	Cactus_Tall_02	30	8.13%	8.33%	0.20%
Vegetation	Cactus_Tall_04	30	8.13%	8.33%	0.20%
Vegetation	Cactus_Leafy_01	30	8.13%	8.33%	0.20%
Trap	Pf_Trap_Cutter	6	20.00%	20.00%	0.00%
Trap	Dynamite	6	20.00%	20.00%	0.00%
Trap	Pf_Trap_Fire	6	20.00%	20.00%	0.00%
Trap	Pf_Trap_Needle	6	20.00%	20.00%	0.00%
Trap	SPIDER	6	20.00%	20.00%	0.00%
HardObject	Chopped_Wood_Pile	2	14.29%	9.09%	5.19%
HardObject	Rock_Large	2	14.29%	9.09%	5.19%
HardObject	Wooden_Barrel	2	14.29%	9.09%	5.19%
HardObject	Rock_Small_02	1	7.14%	9.09%	1.95%
HardObject	Wooden_Barrel (1)	1	7.14%	9.09%	1.95%
HardObject	Rock_Cone	1	7.14%	9.09%	1.95%
HardObject	Rock_Medium	1	7.14%	9.09%	1.95%
HardObject	Rock_Small_01	1	7.14%	9.09%	1.95%
HardObject	Rock_Heavy	1	7.14%	9.09%	1.95%
HardObject	Wooden_Bucket	1	7.14%	9.09%	1.95%
HardObject	Wooden_Crate	1	7.14%	9.09%	1.95%
					0.14%

7.4. Прилог 4 - Подаци о играчима током играња напредне верзије видео игре

Време (секунди)	**Поена сакупљено**	**Обеката сакупљено**	**Поена по секунди**
88.30	1350	119	15.29
55.11	870	111	15.79
51.56	618	82	11.99
9.17	54	13	5.89
116.67	1224	115	10.49
6.23	84	3	13.48
6.97	48	17	6.88
58.27	822	96	14.11
53.09	510	59	9.61
31.88	138	5	4.33
36.03	174	5	4.83
32.93	618	16	18.77
60.29	930	26	15.43
60.10	1326	32	22.06
45.63	618	89	13.54
102.01	1326	113	13.00
17.62	120	23	6.81
42.89	360	54	8.39
42.89	360	54	8.39
33.30	276	60	8.29
29.43	402	69	13.66
46.69	**582**	**55**	**11.48**

Референце

[1] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, Alexandru Iosup, Procedural Content Generation for Games: A Survey, New York: ACM, 2013.

[2] Raúl Lara-Cabrera, Mariela Nogueira-Collazo, Carlos Cotta, Antonio J. Fernández-Leiva, Game Artificial Intelligence: Challenges for the Scientific Community, Málaga: Department "Lenguajes y Ciencias de la Computación", ETSI Informática, University of Málaga, Campus de Teatinos.

[3] A. Amato, Procedural Content Generation in the Game Industry.

[4] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, Cameron Browne, Search-Based Procedural Content Generation: A Taxonomy and Survey, 2011.

[5] Cameron Browne, Simon Colton, Michael Cook, Jeremy Gow, Robin Baumgarten, Towards the adaptive generation of bespoke game content, John Wiley & Sons, Inc., 2012.

[6] T. E. Hull, A.R. Dobell, Random Number Generators, 1962.

[7] „https://mzucker.github.io/html/perlin-noise-math-faq.html,“ [Na mreži].

[8] Jingyuan Huang, Alex Pytel, Cherry Zhang, Stephen Mann, Elodie Fourquet, Marshall Hahn, Kate Kinnear, Michael Lam, William Cowan, An Evaluation of Shape/Split Grammars for Architecture, Waterloo, Ontario, 2009.

[9] B. Lavender, The Zelda Dungeon Generator: Adopting Generative Grammars to Create Levels for Action-Adventure Games, 2015.

[10] A. Alattas, Procedural developing of a reconstruction of archaeological model Applied to the case study: Nymphaeum of the “sanduary di Diana” in Nemi, Italy, 2014.

[11] „Pythagoras tree (fractal),“ [Na mreži]. Available: https://en.wikipedia.org/wiki/Pythagoras_tree_(fractal).

[12] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, Steven Worley, Texturing and modeling: A Procedural Approach, Academic Press Ltd..

[13] „Computing Classification System, 2012 Revision,“ 2012. [Na mreži]. Available: https://www.acm.org/publications/class-2012.

[14] S. Wolfram, Statistical mechanicsof cellular automata.

[15] J. Gips, Shape Grammars and their Uses: Artificial Perception, Shape Generation and Computer Aesthetics, 1975.

[16] B. M. Namee, Agent Based Modeling in Computer Graphics and Games, 2009.

[17] Aung Sithu Kyaw, Clifford Peters, Thet Naing Swe, Unity 4.x Game AI Programming, Birmingham: Packt Publishing Ltd., 2013.

[18] Xinjie Yu, Mitsuo Gen, Introduction to Evolutionary Algorithms, Springer, 2010.

[19] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, Springer, 2015.

[20] M. Melanie, An Introduction to Genetic Algorithms, A Bradford Book The MIT Press, 1999.

[21] Georgios N. Yannakakis, Julian Togelius, Experience-Driven Procedural Content Generation, 2011.