Univerzita Komenského
Fakulta Matematiky, Fyziky a Informatiky
Ústav informatiky

Mária Šarkanová

Použitie evolučných algoritmov v interiérovom dizajne

Diplomová práca
Diplomový vedúci: Mgr. Ján Habdák

Bratislava 2004

Týmto prehlasujem, že som diplomovú prácu vypracovala samostatne s odbornou pomocou školiteľa.

Bratislava, apríl, 2004 Mária Šarkanová

Poďakovanie.

Ďakujem Mgr. Jánovi Habdákovi za cenné rady a pripomienky pri písaní tejto diplomovej práce.

Obsah

Kapitola 1

Úvod

Na modelovanie objektov reálneho sveta a na manipuláciu s nimi existuje viacero možných prístupov. Medzi ne patria aj genetické algoritmy, ktoré boli podobne ako napríklad neurónové siete, inšpirované prírodou.

Jedným z nových trendov v použití genetických algoritmov je v súčasnosti navrhovanie systémov pre evolučný dizajn. Ich cieľom je zlepšiť produktivitu práce dizajnéra a znížiť výdavky pri vytváraní dizajnu. Medzi hlavné úlohy evolučného dizajnu patrí generovanie nových riešení za pomoci dizajnéra. Cieľom môže byť aj samostatné navrhnutie úplne nového dizajnu.

Evolučné algoritmy sú na tento účel vhodné najmä preto, že sú schopné samostatne kreatívne vyvíjať objekty, ktorých evolúcia nie je explicitne formulovaná v evolučnom algoritme. Táto vlastnosť je emergentnou vlastnosťou evolučného algoritmu. Emergentná vlastnosť nie je napevno naprogramovaná, ale vzniká sekundárne pri behu algoritmu z iných vlastností. (Napríklad objekt, ktorý je symetrický podľa dvoch vhodných rovín je okrem jedného prípadu automaticky stabilným objektom)

Evolučný dizajn sa môže zaoberať riešením viacerých druhov problémov. Peter Bentley navrhuje vo svojej knihe Evolutionary Design by Computers nasledovné delenie problémov pre evolučný dizajn:

- optimalizácia už existujúceho dizajnu
- kreatívny evolučný dizajn - generovanie úplne nového dizajnu z náhodnej počiatočnej populácie
- evolučné generovanie umeleckých diel

- vývoj umelých foriem života

Tieto oblasti sa líšia navzájom hlavne v spôsobe reprezentácie problému a vo vytváraní počiatočnej populácie.

Vynára sa otázka: Prečo evolvovať dizajn pomocou počítača? Existujú mnohé dôvody, prečo je evolvovanie dizajnu za pomoci počítača vhodným spôsobom riešenia problému. Evolúcia je všeobecný návod na riešenie ľubovoľného problému a evolučné algoritmy boli úspešne použité pri riešení veľkého počtu problémov. Taktiež proces evolúcie a postup dizajnéra pri návrhu sú veľmi podobné a veľa pozoruhodných a aj veľmi úspešných dizajnov bolo vyvinutých v prírode, ktorá je inšpiráciou pre evolučné algoritmy [1].

1.1 Cieľ práce

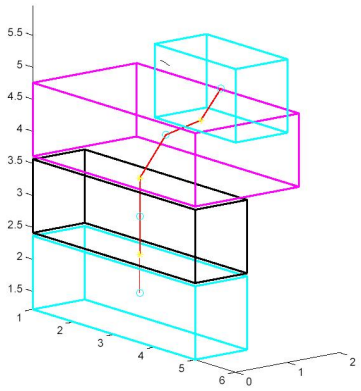
Cieľom mojej práce je navrhnúť reprezentáciu objektu, konkrétne stoličky, pre genetický algoritmus a implementovať genetický algoritmus používajúci danú reprezentáciu, pomocou ktorého sa pokúsim vyvinúť stoličku, ktorá bude použiteľná aj v bežnom živote. Navrhnúť estetické kritériá pre ohodnotenie stoličky nie je mojim cieľom.

1.2 Prehľad doterajších riešení

Evolučné algoritmy ako optimalizačné metódy boli už použité veľakrát a úspešne na optimalizovanie rôznych problémov. V súčasnosti sa však do pozornosti dostáva nová, zatiaľ dobre nepreskúmaná, oblasť evolučného dizajnu. Pomocou evolučného dizajnu boli navrhované viaceré interiérové doplnky a aj zariadenia vo všeobecnosti ako napríklad žeriav.

1.2.1 Návrh Legového stola a žeriavu

P.J. Funes a B. Pollack sa vo svojej práci [2] zaoberali vytváraním objektov z vopred definovaných stavebných prvkov (Legové kocky). Nosnými časťami ich reprezentácie problému boli rôzne druhy kociek, ich vlastnosti a spojenia medzi kockami. Pre každé spojenie medzi dvoma kockami je daná sila, ktorú treba vynaložiť na to, aby sa tieto dve kocky od seba oddelili. Každá kocka má nejakú váhu a môžu k nej byť pripojené ďalšie kocky, ktoré jej pridávajú na váhe. Aby spoj medzi dvoma kockami bol stabilný, musí platiť, že súčet všetkých síl, ktoré pôsobia na toto spojenie, je menší ako sila, ktorá ich drží dokopy. Týmto sa váha kociek, ktoré sú už stabilne spojené, propaguje



Obrázok 1.1: ilustrácia reprezentácie

ďalej. Ak sa takýmto spôsobom dá prejsť celá štruktúra a výsledná sila je absorbovaná pevnou podložkou, je takáto štruktúra stabilná.

Túto štruktúru reprezentovali pomocou stromu, pričom každý vrchol obsahoval nejakú kocku a zoznam nasledovníkov, ktorí môžu byť k tejto kocke pripojení. Každý takýto nasledovník má 3 parametre, ktoré popisujú relatívne jeho súradnice vzhľadom na predchádzajúcu kocku, číslo udávajúce otočenie vzhľadom na pôvodnú kocku a samotnú veľkosť tejto kocky.

Operátor mutácie robí náhodnú zmenu parametrov kocky - veľkosť, pozícia, orientácia, alebo pridanie novej kocky.

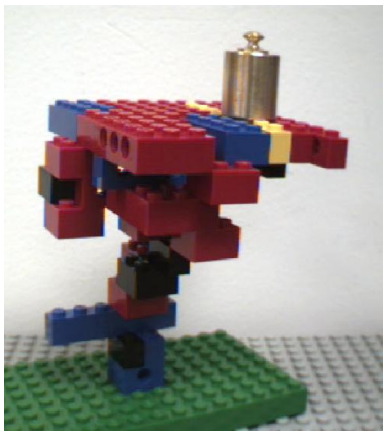
Operátor kríženia vyberie dva náhodné podstromy z rodičovských stromov a v potomkovi nahradí jeden podstrom druhým podstromom.

Vyskytol sa problém s tým, že takto vzniknutý strom nemusí byť korektný (niektoré kocky by sa mohli prekrývať). Riešili ho tak, že keď sa takáto situácia po krížení objaví, tak sa výsledný strom oreže v mieste, kde došlo ku kolízii. Touto procedúrou získajú maximálny korektný - postaviteľný strom.

Návrh stola

Pomocou takejto reprezentácie sa pokúsili vyvinúť stôl. Kritériá pre dobrý stôl boli nasledovné:

- Stôl musí mať nejakú požadovanú výšku.
- Povrch stola musí pokrývať danú plochu.
- Stôl musí zostať stabilný, aj keď sa na ľubovoľné miesto na povrchu stola položí dodatočné závažie.



Obrázok 1.2: vyvinutý stôl

- Ak sú splnené ostatné kritériá, preferovaný je nižší počet kociek.

Fitness funkcia bola definovaná tak, že za splnenie prvej podmienky dávala 1 až 2 body, za splnenie prvej a čiastočne splnenie druhej podmienky 2 až 3 body atď. S takto definovanou fitness funkciou algoritmus najprv ťahal štruktúru do požadovanej výšky, potom rozširoval plochu stola, zaisťoval, že všetky body plochy udržia závažie a nakoniec sa snažil redukovať počet kociek.

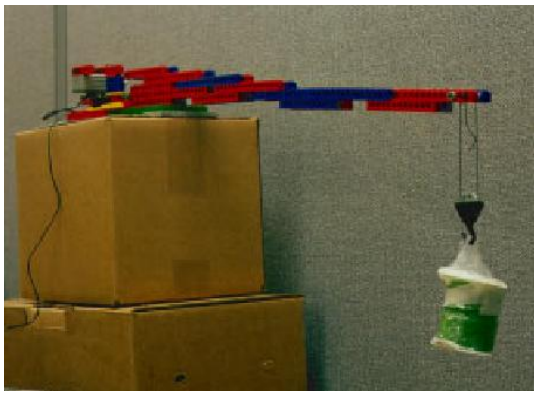
Problémy s touto reprezentáciou bol napríklad v tom, že vzdialenosť medzi genotypom a fenotypom je veľká, čo spôsobuje, že mutácia je vo väčšine prípadov príliš radikálnou zmenou. Vyskytol sa aj problém s fitness funkciou, kde pri prechode medzi hodnotami 1,9 a 2,0 bol príliš malý selektívny tlak na zlepšovanie riešení. Tento problém riešili tak, fitness funkciu zmenili na exponenciálnu funkciu, čím sa zvýraznili predtým malé rozdiely.

Návrh ramena žeriavu

Fitness funkcia oceňovala podobným spôsobom ako v predchádzajúcom prípade:

- Dosiahnutie vopred zadanej vzdialenosti od podložky.

- Ak bola táto vzdialenosť dosiahnutá, tak fitness funkcia udelila body naviac za to, že na poslednú kocku sa dalo zavesiť nejaké závažie.



Obrázok 1.3: vyvinuté rameno žeriavu

1.2.2 Návrh stola pomocou Lindenmayerových systémov

G.S. Hornby a J.B. Pollack v tejto práci [3] používajú Lindenmayerove systémy na kódovanie genotypu. V genotype sú týmto spôsobom zakódované príkazy, podľa ktorých sa má zostrojiť výsledný objekt.

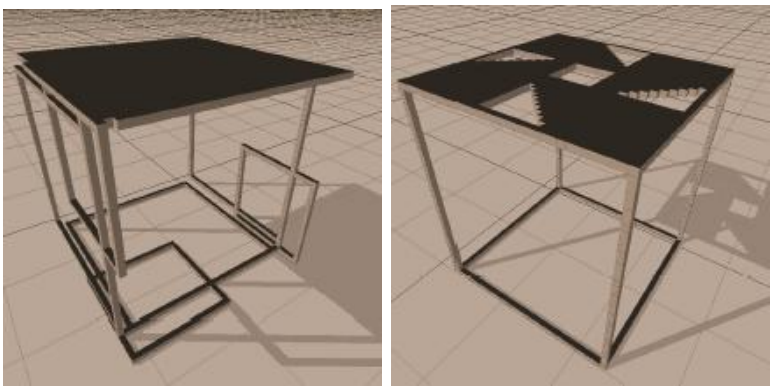
Na stavbu objektov používajú kocky rovnakej veľkosti. Pravidlá na vytváranie objektu sú sformulovane podobne ako príkazy pre 3D korytnačku, napríklad: choď dopredu, choď dozadu, otoč sa do nejakého smeru. Pri pohybe dopredu korytnačka uloží na miesto, kde sa nachádza kocku, pri pohybe naspäť kocku odstráni.

Evolučný algoritmus sa v tomto prípade používa na evolúciu L-systému, podľa ktorého sa potom vytvára výsledný objekt. Každý objekt môže mať len obmedzené množstvo pravidiel. Tieto pravidla sú na začiatku vygenerované náhodne.

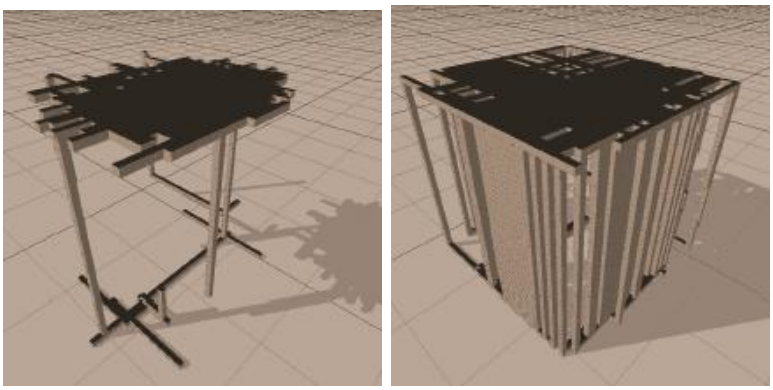
Mutácia je definovaná ako malá zmena niektorého z pravidiel. Táto zmena môže byť zmena príkazu na iný príkaz, malá zmena parametra nejakého príkazu, zmena podmienky na vykonanie pravidla, pridanie/odobratie nejakého príkazu alebo zapúzdrenie nejakého bloku príkazov a vyrobenie nového pravidla z tohto bloku.

Operátor kríženia zoberie dvoch rodičov a potomka vyrobí tým, že skopíruje jedného z rodičov a potom nahradí nejakú časť pravidla časťou pravidla druhého rodiča.

Fitness funkcia oceňuje tieto vlastnosti: výška stola (výška najvyššie položenej kocky), počet kociek v najvyššej vrstve (doska stola). Fitness funkcia tiež preferuje stoly, ktoré majú menej kociek, okrem kociek tvoriacich dosku stola.



Obrázok 1.4: Stoly vyvinuté s použitím generatívnej reprezentácie

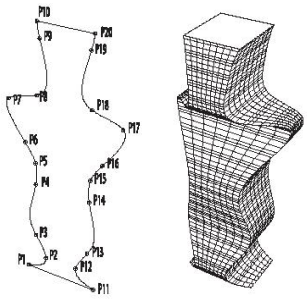


Obrázok 1.5: Stoly vyvinuté s použitím negeneratívnej reprezentácie

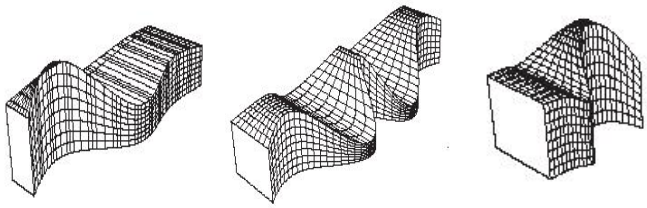
Takáto reprezentácia problému je príkladom generatívnej reprezentácie. Vo svojej práci experimentovali aj s použitím negeneratívnej reprezentácie na riešenie tej istej úlohy. Hlavným rozdielom medzi týmito dvoma prístupmi bol fakt, že pri generatívnej reprezentácii je možné využiť viackrát už vyvinuté dobré časti. Túto situáciu pekne ilustruje obrázok 1.4.

1.2.3 Návrh nábytku, ktorý sa dá zostrojiť z jedného kusu materiálu

Nábytok, ktorý sa spomína v tejto práci sa vyrába tak, že na plochý materiál, z ktorého bude objekt vytvorený, sa nakreslia línie, pozdĺž ktorých sa tento materiál ohne, čím vznikne požadovaný priestorový útvar. Marcus Schein vo svojej práci [4] používa jednoduchú geometrickú reprezentáciu objektu. Ob-



Obrázok 1.6: Príklad zakódovania objektu a výsledného objektu



Obrázok 1.7: Vyvinuté ležadlo, ležadlo pre dve osoby a stolička

jekty, ktoré vyvíja sú reprezentované pomocou splajnov, ktoré určujú, ako sa má poskladať materiál na výrobu objektu. Tieto splajny sú zakódované v chromozómoch pomocou dvojrozmerných súradníc bodov, ktorými splajn prechádza a počtom týchto bodov. Používateľ má možnosť nastavovať základné parametre objektu - minimálna a maximálna výška a šírka, použitie symetrie a počet bodov v splajne. Pomocou rôzneho počtu bodov sa dá dosiahnuť, že vyvinutý objekt môže mať rôzne použitia. Keď sa použije málo bodov, objekt pripomína stôl, keď sa použije viac bodov, objekt je použiteľný ako stolička, ešte viac bodov spôsobí, že vyvinutý objekt môže slúžiť ako kreslo pre dvoch atď.

1.2.4 Použitie evolučných algoritmov v architektúre

Evolučné algoritmy boli úspešne použité aj v architektúre, kde sa uplatnili pri navrhovaní pôdorysov budov spĺňajúcich nejaké kritériá [5], alebo na evolúciu budov v nejakom štýle, ktorý bol predtým matematicky popísaný. V súčasnosti sa pracuje na vypracovaní podobných popisov na rôzne špecifické druhy budov, napríklad čínske pagody, tradične bosnianske domy [6] a pod. čo umožní použitie evolučných algoritmov na navrhovanie takýchto budov.

Kapitola 2

Základné pojmy

2.1 Evolučný algoritmus

Evolučné algoritmy sú optimalizačné techniky založené na princípoch evolúcie. Evolučný algoritmus pracuje s populáciou chromozómov. V každom chromozóme je zakódované nejaké riešenie problému. Podľa informácií zakódovaných v chromozóme (genotyp) sa vytvorí konkrétne riešenie (fenotyp). Kvalita ľubovoľného riešenia sa dá ohodnotiť pomocou fitness funkcie. Zakódované riešenia musia byť pred tým, ako sa vyhodnotí fitness každého riešenia, namapované na príslušné fenotypy. Prostredníctvom prirodzenej selekcie a genetických operátorov kríženia a mutácie sa hľadajú chromozómy s lepšou hodnotou fitness. Táto prirodzená selekcia zaručuje, že chromozómy s najvyššou fitness sa budú šíriť v nasledujúcich generáciách. Pri krížení sa kombinujú rodičovské chromozómy a vytvárajú sa z nich nové chromozómy potomkov, ktoré majú vysokú pravdepodobnosť, že ich fitness bude lepšia ako fitness rodičov. Mutácia zabezpečuje vznik nových vlastností, ktoré sa nedajú zdediť od rodičov. Genetický algoritmus sa pokúša o postupné zvyšovanie fitness chromozómov v populácii a po mnohých generáciách vytvorí chromozómy, ktoré obsahujú optimalizované premenné.

Podrobnejšie informácie o evolučných algoritmoch a ich ďalšom delení sa dajú nájsť v [7].

2.2 Reprezentácia problému

Aby sme mohli riešiť ľubovoľný problém pomocou evolučných algoritmov, musíme nájsť spôsob, ako riešenia kódovať v chromozómoch. Zvolený spôsob kódovania ovplyvní tvar riešení. Ak sa napríklad v reprezentácii nedá zakódovať nejaká podmnožina riešení, tieto sa nebudú dať vyvinúť, pretože

vyvinúť sa dajú len riešenia, ktoré sa dajú reprezentovať. Preto je veľmi dôležité navrhnúť reprezentáciu tak, aby pokrývala čo najväčšiu množinu riešení.

Reprezentácia je dôležitá aj pri riešení zložitých problémov. Popísať presne všetky vlastnosti riešení takýchto zložitých problémov môže byť nemožné. Preto je dôležité vyselektovať podstatné vlastnosti riešení, ktoré očakávame. Pri ďalšom riešení problému sa potom zaoberáme len týmito vybranými vlastnosťami a od iných charakteristík reálnych objektov abstrahujeme.

Pri voľbe reprezentácie problému pre genetický algoritmus sa treba zaoberať viacerými otázkami ohľadne typu reprezentácie a samotného reprezentovania objektu zvoleným typom reprezentácie.

2.2.1 Voľba typu reprezentácie

Reprezentácia problému pre genetický algoritmus môže byť dvoch typov:

- *Statická:* Každý jedinec v populácii má rovnako dlhý chromozóm a na ňom rovnaké gény, jedince v populácii sa líšia obsahom génov.
- *Dynamická:* Dĺžka genetickej reprezentácie jedincov v populácii sa môže líšiť. Zložitejšie jedince majú dlhšiu genetickú reprezentáciu

Statická reprezentácia je určená na riešenie problémov, v ktorých má riešenie vždy rovnakú množinu vlastností. Tieto riešenia môžeme kódovať tak, že každej vlastnosti priradíme jeden gén na chromozóme. Všetky chromozómy majú potom rovnakú dĺžku. Ale ak máme problém, ktorého riešenie sa nedá exaktne popísať, tak je problematické navrhnúť takúto statickú reprezentáciu. Preto sa používa dynamická reprezentácia, kde gény na chromozómoch nepopisujú vlastnosti riešenia ale dávajú návod, ako toto riešenie zostrojiť. Takáto reprezentácia má často formu stromu, lebo komplexné útvary by sa ťažko reprezentovali v lineárnej forme. Stromová štruktúra ma výhodu aj v tom, že sa na nej dá pomerne jednoducho zadefinovať operátor kríženia ako vzájomná výmena náhodných podstromov reprezentácie dvoch jedincov.

2.2.2 Samotná voľba reprezentovania komponentov v reprezentácii problému

Existujú dva základné druhy popisu komponentov:

- *Generatívny*
- *Popisný (negeneratívny)*

Pri generatívnom popise je charakteristické, že reprezentácia je tvorená príkazmi nejakého programu, ktorý skonštruuje výsledný objekt. Hornby a Pollack pri riešení problému evolučného dizajnu - návrhu stola [3], využili pri reprezentovaní problému Lindenmayerove systémy. V ich prístupe sa pri evolúcii menili pravidlá, podľa ktorých sa potom vytvoril objekt. Takéto použitie L-systémov ako generatívnej reprezentácie má veľkú výhodu v tom, že umožňuje kompaktnejšiu reprezentáciu riešenia problému, lebo umožňuje znovu použiť už vyvinuté časti genotypu.

Negeneratívny popis je jednoduché popísanie nasledujúceho kroku pri konštrukcii objektu. Napríklad pri evolúcii Legových štruktúr v [2], je reprezentácia tvorená stromom, pričom každý vrchol stromu popisuje jednu kocku Lega a jeho potomkami sú všetky ďalšie kocky, ktoré sú k tejto kocke fyzicky pripojené. Každý uzol v strome má taktiež relatívny popis polohy voči predchádzajúcej kocke.

Kapitola 3

Moje riešenie

3.1 Úvodné úvahy

Na začiatku mojich úvah o probléme evolúcie stoličiek bolo rozhodnutie o tom, či budem objekty, s ktorými pracujem, aj vykresľovať. Druhá možnosť je, že budem s nimi manipulovať bez toho, aby som mohla vidieť, ako v skutočnosti vyzerajú. Tento prístup by bolo možné použiť v prípade, keby mojim cieľom bolo napríklad vylaďovanie hodnôt jednotlivých parametrov kritérií a podobne. Pri takejto práci nie dôležité, ako presne objekty v skutočnosti vyzerajú. Ak by som už mala identifikované dôležité kritériá, ktoré majú spĺňať výsledné objekty, tak v tomto prípade je relevantné sledovanie vývinu hodnôt fitness funkcie.

Ale v mojom prípade som ešte nemala identifikované potrebné kritériá. Aby som mohla tieto kritéria sformulovať a otestovať, som nutne potrebovala vidieť objekty, ktoré boli výsledkom evolúcie. Potom som mohla zhodnotiť, či je navrhnuté kritérium dobré, prípadne zistiť, že ešte to nie je celkom v poriadku a ešte treba niečo vymyslieť. Preto bolo dôležitou časťou mojej práce nájsť spôsob, ako vizualizovať objekty, s ktorými pracujem.

Po preskúmaní viacerých možností som sa rozhodla použiť OpenGL, ktoré je v súčasnosti jednou z najlepších grafických knižníc. Pre OpenGL je väčšina manuálov, ktoré sú k dispozícii, písaná pre programovací jazyk C++. Keďže nie som odborník v oblasti počítačovej grafiky, použila som vo svojej práci práve tento programovací jazyk. Začala som teda tým, že som si pripravila základný predpoklad - program schopný vizualizovať ľubovoľný jednoduchý útvar. Potom som sa mohla začať uvažovať o samotnom evolučnom algoritme a reprezentácii problému.

3.2 Reprezentácia

Ďalším krokom bol návrh reprezentácie problému. Najprv som sa mala rozhodnúť, či použijem reprezentáciu statickú alebo dynamickú. Dôvodom použitia dynamickej reprezentácie je, že objekty, ktorými sa zaoberám, sa nedajú popísať jednoducho množinou svojich vlastností.

Čo sa týka generatívnosti reprezentácie, bolo by možné použiť L-systémy alebo podobný systém ako generatívnu reprezentáciu problému. Podobným problémom sa zaoberali už iní ľudia [3], ja som sa týmto spôsobom reprezentácie nechcela zaoberať aj preto, lebo som chcela vyskúšať nejaký nový prístup. Ďalším problémom by asi bola zložitosť implementácie takéhoto systému. Programovací jazyk C++ totiž neposkytuje dobré prostredie na manipuláciu s reťazcami a tiež použitie grafickej knižnice OpenGL v iných programovacích jazykoch by mohlo byť tiež dosť problematické. Preto som sa rozhodla použiť inú, negeneratívnu reprezentáciu.

3.2.1 Voľba komponentov reprezentácie

Nasledovalo rozhodnutie, z akých komponentov sa budú skladať stoličky v mojej reprezentácii. Rozhodla som sa pre čiary a plochy, plochy sú definované pomocou čiar, ktoré ich ohraničujú. Čiary sú podľa mňa univerzálny útvar, pomocou ktorého sa dá reprezentovať ľubovoľný objekt. Plochy sú dôležité preto, lebo pomocou nich sa dá modelovať kľúčová časť každej stoličky - sedadlo.

Reprezentovanie štruktúry objektu

Vo svojich úvahách som predpokladala, že stolička je súvislý objekt, a preto sa dajú navzájom pospájané úsečky reprezentovať pomocou nejakého zoznamu. Ako ale zachytiť v reprezentácii, akým spôsobom sú úsečky navzájom pospájané? Prvým prístupom, nad ktorým som uvažovala, je reprezentácia pomocou grafu. Tento graf tvoria vrcholy, ktoré reprezentujú jednotlivé úsečky a vrcholy patriace úsečkám, ktoré spolu susedia, sú spojené hranou. Toto riešenie sa ukázalo ako nevhodné, lebo nebolo možné dosť dobre zadefinovať kríženie na takejto štruktúre. Hlavným problémom bolo, akým spôsobom oddeliť nejakú časť grafu od zvyšku grafu, ak sú tieto dve časti prepojené viacerými hranami. Druhým vážnym problémom bolo, že keby sa nám aj podarilo nejakým rozumným spôsobom oddeliť časť grafu, tak by bolo ešte ťažšie ju nejako konzistentne vložiť do druhého grafu.

Pre tieto problémy som sa rozhodla použiť taký graf, v ktorom nie sú kružnice, spôsobujúce problémy pri predchádzajúcom prístupe. Pri stromo-

vej reprezentácii sa dá kríženie zadefinovať veľmi jednoducho ako vzájomná výmena náhodných podstromov. Toto zjednodušenie spôsobilo obmedzenie množstva útvarov, ktoré sa dajú reprezentovať, len minimálne. Nedajú sa reprezentovať útvary, ktoré v sebe obsahujú uzavretý sled úsečiek. Uzavretý sled úsečiek sa však dá nasimulovať tak, že prvý bod prvej úsečky sledu je rovnaký ako posledný bod poslednej úsečky. Teda ani takéto štruktúry nie sú teoreticky vylúčené, len je veľmi nepravdepodobné, že by vznikli náhodne.

Keď už som mala vymyslené, ako budem zaznamenávať súvislosti medzi úsečkami, zostávalo ešte vyriešiť detaily ako napríklad kedy je povolené vetvenie v strome, koľkonásobné toto vetvenie môže byť a podobne.

Reprezentovanie úsečiek

Reprezentácie úsečiek by mala v každom prípade mať takú vlastnosť, že jedna úsečka ovplyvňuje umiestnenie len úsečiek s ňou priamo susediacich. To znamená, že ak pridám jednu úsečku napríklad na začiatok nejakého útvaru, tak sa nezmení tvar pôvodného útvaru. Môže sa však zmeniť jeho orientácia v priestore, pretože to závisí od orientácie pridanej úsečky. Takýto spôsob interpretácie údajov v chromozómoch je dôležitý na to, aby nedochádzalo k deformovaniu úsečiek, ktoré nie sú menené.

Prakticky sa snažím zachovávať uhol medzi dvomi za sebou nasledujúcimi úsečkami. Zobrazenie, ktoré zabezpečuje túto vlastnosť je otáčanie priestoru okolo osi kolmej na obidve úsečky. Problémom je, že priestor takýchto zobrazení nie je spojitý. V dvoch bodoch nespojitosti som radšej definovala iné zobrazenia. Táto situácia nastáva vtedy, ak obidve úsečky ležia na tej istej priamke. V tomto prípade totiž existuje nekonečne veľa smerov kolmých vektorov.

Dôvod, prečo chcem, aby bol priestor zobrazení spojitý je v tom, že považujem za vhodné, aby malá zmena smeru vektora spôsobila malú zmenu odchýlky.

Reprezentovanie plôch

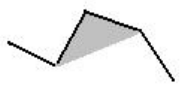
Problémy, ktoré sa vyskytli pri pokuse zahrnúť do reprezentácie plochy, boli rôzne. Jedným z problémov bolo, akým spôsobom plochu reprezentovať. Ďalším problémom bolo, akým spôsobom bude treba pozmeniť operáciu kríženia, aby vedela pracovať aj s útvarom s plochami.

Plochy je možné do reprezentácie zahrnúť viacerými spôsobmi.

Prvým spôsobom, ako zahrnúť plochy do reprezentácie, bol nasledovný prístup. Plocha by sa mohla nachádzať v ľubovoľnom uhle medzi dvoma

úsečkami. Tieto dve úsečky by tvorili dve z ramien trojuholníka. Informácia o prítomnosti plochy by sa mohla nachádzať v spojovacom bode medzi úsečkami. Tento spôsob vytvárania plôch neumožňoval súvislo vyplniť väčšiu plochu.

Pri krížení mohla vzniknúť situácia, že by sa mal objekt rozdeliť v uzle, v ktorom sa nachádza plocha. Riešením by mohlo byť zakázanie kríženia v týchto bodoch. To by mohlo viesť k tomu, že sa v stoličke budú nachádzať dlhé úseky plôch, ktoré nebude možné rozdeliť za účelom kríženia. Druhým riešením by mohlo byť, že na plochu sa pri krížení neberie ohľad. To znamená, že plocha by mohla v tomto prípade zaniknúť, alebo by sa spoľahla na to, že sa pri krížení dostane na miesto chýbajúcej úsečky iná úsečka.



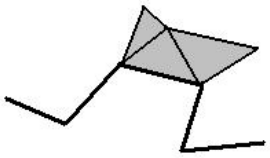
Obrázok 3.1: Ilustrácia prvého spôsobu zahrnutia plôch do reprezentácie. Rovina nachádzajúcej sa v uhle medzi dvoma úsečkami.

Tento spôsob som zavrhla, lebo sa nedajú vytvoriť vyplnené väčšie útvary, napríklad päťuholník.

Druhým spôsobom, nad ktorým som uvažovala, bol pridávanie trojuholníkových plôch na úsečky. Ku každej úsečke mohol byť priradený trojuholník, ktorého poloha bola určená ďalšou úsečkou. Tretia úsečka trojuholníka by sa dopočítavala tak, aby bol útvar uzavretý. Na každej z týchto dvoch novovzniknutých úsečiek mohol byť ďalší trojuholník.

Hlavný problém tejto reprezentácie je v tom, že všetky väčšie plochy, ktoré vyrába, sú vo väčšine prípadov nekonvexné. Väčšie súvislé plochy, napríklad päťuholníky sa síce dajú vyrobiť, ale je veľmi nepravdepodobné, že by vznikli náhodou. Bolo by treba aj zvážiť možnosť vetvenia na tom vrchole trojuholníka, ktorý neleží na pôvodnej úsečke.

Tretí spôsob som použila v mojej reprezentácii, lebo som ho považovala za vyhovujúci. Plocha je v tomto prípade definovaná postupnosťou úsečiek, ktoré ju ohraničujú. Posledná úsečka je dopočítaná tak, aby posledný bod a prvý bod boli totožné. Týmto spôsobom je teoreticky možné vyrábať ľubovoľne veľké plochy. Problémom je ale, že takáto všeobecná plocha nebude



Obrázok 3.2: Ilustrácia druhého prístupu k zahrnutiu plôch do reprezentácie. Trojuholníkové roviny sa nachádzajú na úsečkách.

rovinou. Pri skúmaní spôsobu, akým by sa dali takéto všeobecné plochy narovnať, som narazila na matematický problém, ako to realizovať. Tento problém sa mi nezdal podstatný a jeho riešenie som odložila na neskôr, ale potom som sa k nemu už nevrátila.

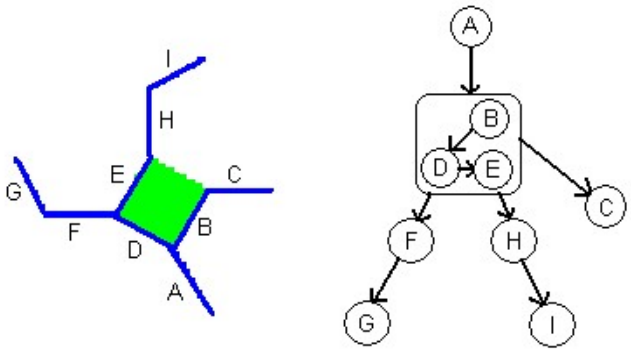
Úsečky ohraničujúce plochu nie sú plnohodnotné úsečky. To znamená, že sa na nich nemôže nachádzať ďalšia plocha a nedajú sa krížiť. S rovinou sa manipuluje ako s celkom. Operátor kríženia na takéto roviny nemusí brať ohľad, pretože rovina zostane na úsečke, ku ktorej je pridružená. Vetvenie môže nastať v každom vrchole roviny, teda v každom vrchole roviny sa môže nachádzať ďalšia plnohodnotná úsečka.

3.2.2 Presný popis reprezentácie

Základnými časťami objektov v mojej reprezentácii sú úsečky a roviny. Na reprezentáciu objektu používam stromovú štruktúru, pričom každý vrchol stromu obsahuje informáciu o jednej úsečke. Popis úsečky zahŕňa:

- zmenu orientácie úsečky v priestore vzhľadom na predchádzajúcu úsečku
- dĺžku úsečky
- informáciu o tom, či je k tejto úsečke pridružená rovina. Ak áno, tak vrchol obsahuje aj informáciu o úsečkách, ktoré ohraničujú rovinu.

Ak k úsečke nie je priradená rovina, tak v tom vrchole nedochádza k vetveniu, ale môže mať jedného potomka - nasledujúcu úsečku. Ak k úsečke je priradená nejaká rovina, tak pre každý vrchol roviny môže mať tento uzol stromu jedného potomka. Teda k stromovému vetveniu dochádza vtedy, keď sa vo vrchole okrem úsečky nachádza ešte aj rovina pripojená k tejto úsečke.



Obrázok 3.3: Príklad objektu a jeho reprezentácie. Úsečky sú označené písmenami, plocha je šedá.

Kríženie dvoch stoličiek je definované ako vzájomná výmena ľubovoľných podstromov.

Obrázok 3.4: Rodič 1 a rodič 2, kruhom je vyznačené miesto kríženia

Obrázok 3.5: Potomok 1 a potomok 2

Mutácia čiary je definovaná ako zmena niektorého z parametrov v strome a teda je to vlastne zmena orientácie niektorej úsečky a v súvislosti s tým aj celeho zvyšku stoličky.

Obrázok 3.6: Príklad mutácie objektu

3.3 Ako prevádzam genotyp na fenotyp

Na to, aby som mohla vyhodnocovať fitness funkciu a vykresľovať stoličku, potrebujem poznať reálne súradnice všetkých bodov stoličky. Preto potrebujem zadefinovať nejakú transformáciu, ktorá bude hovoriť o tom, ako sa prevádzajú údaje zakódované v genotype na reálne súradnice. Keďže v mojej reprezentácii sú všetky informácie zakódované relatívne vzhľadom na predchádzajúci útvar, potrebujem mať nejaký pevný začiatok.

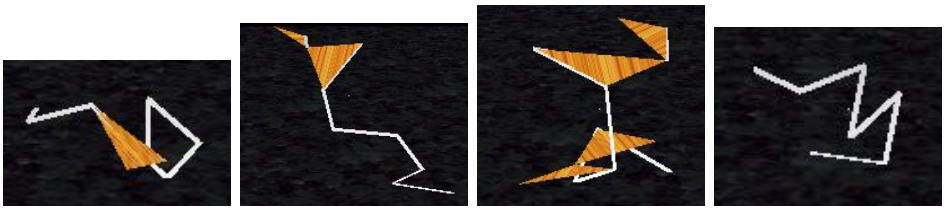
Každá stolička má zadaný nejaký pevný bod v priestore - začiatok. Vzhľadom na tento bod je definovaný inicializačný vektor v smere zápornej x-ovej osi, vzhľadom na ktorý sú prevádzané transformácie zakódované v strome, ktorý reprezentuje nejakú stoličku. Pri vytváraní fenotypu postupne prechádzam celým stromom, spracovávam každú úsečku v reprezentácii stoličky a vypočítavam reálne súradnice úsečiek a rovín v trojrozmernom priestore. Tieto súradnice sú potom používané na výpočet fitness funkcie a na vykresľovanie objektu.

3.4 Ako som postupovala

Pri svojej práci som postupne vymýšľala a pridávala nové kritériá pre fitness funkciu a sledovala som, ako rôzne nastavenia vplývajú na evolúciu objektov.

Tento postup je veľmi náročný z implementačného hľadiska. Môže sa totiž stať, že nové kritérium bude vyžadovať neočakávanú zmenu dátových štruktúr, ktoré sa v programe používajú. Tieto dátové štruktúry môžu byť dosť zložité a takáto zmena potom vyžaduje zdĺhavú prácu pri prerábaní programu tak, aby pracoval s novými štruktúrami. Pri mojej práci sa tento problém objavil niekoľkokrát.

Výhodou je, že tieto zmeny nemajú vplyv na nijaké výpočty, ktoré sa realizujú na fenotype, ako napríklad výpočet hodnoty fitness funkcie a pod. Teda vždy som prerábala len nejakú malú časť programu.



Obrázok 3.7: Príklad stoličiek v náhodne vygenerovanej prvej populácii.

3.4.1 Základné kritériá

Za základné kritérium, ktoré musí spĺňať každý použiteľný objekt, považujem stabilitu objektu.

Stabilita jednoduchých objektov

Na začiatku som stoličku, resp. útvary, na ktorých som testovala vhodnosť kritérií, mala reprezentovanú len ako jednu náhodnú lomenú čiaru. Na tejto reprezentácii som testovala kritérium stability tak, že som oceňovala každý bod, ktorý sa nachádzal na zemi. Rovina zeme bola definovaná ako vodorovná rovina prechádzajúca najnižším bodom útvaru.

Výsledkom evolúcie boli po 30 generáciách takmer samé také útvary, ktoré sa nachádzali v rovine dlážky. Teda tento typ reprezentácie sa ukázal ako celkom dobrý, čo sa týka vytvárania stabilných objektov. Tento prístup, ktorý oceňoval počet bodov na zemi, však preferoval väčšie útvary, pretože mohli mať viacej bodov v rovine dlážky. Objekty pri evolúcii preto rástli do obrovských rozmerov. Tomuto som zabránila tým, že som do fitness funkcie začlenila penalizáciu za prekročenie nejakého limitu na veľkosť objektu. Táto penalizácia bola robená tak, že za každú úsečku nad stanovený limit, bola hodnota fitness funkcie znížená o nejakú malú hodnotu. Táto úprava fitness funkcie zabezpečila, že objekty sa dovyvíjali do nejakej maximálnej veľkosti a počas ďalšej evolúcie už ďalej nerástli. Ich výsledná veľkosť závisela od konkrétnej hodnoty, ktorou boli oceňované body na zemi a od hodnoty penalizácie.

Stabilita zložitejších objektov

Stabilita objektu bola tiež prvým kritériom, ktoré som použila pri testovaní evolúcie s kompletnou reprezentáciou, teda už aj s rovinami. Vyhodnocuje sa tak, že sa vypočíta ťažisko celého objektu. Potom sa zistia oporné body objektu, t.j. body, ktoré sa dotýkajú zeme, a vyhodnotí sa, či sa ťažisko

nachádza nad konvexným obalom oporných bodov v rovine podlahy. Rovina podlahy je vodorovná rovina prechádzajúca najnižším bodom stoličky.

Asi najväčší problém, s ktorým som sa stretla pri pokusoch so stabilitou objektov, bol fakt, že je veľmi nepravdepodobné, že náhodne vytvorený objekt bude stabilný. Pri veľkosti populácie 100 sa často stávalo, že ani jeden objekt nespĺňal kritérium stability a ani sa ho v priebehu pár desiatok generácií nepodarilo vyvinúť, ak fitness funkcia iba oceňovala ľubovoľný stabilný objekt. Ak sa nejaký stabilný objekt v priebehu evolúcie objavil, potom bol preferovaný pri vytváraní potomkov. Keďže s najväčšou pravdepodobnosťou bude stabilný potomok len ten, ktorý zdedil stabilnú časť od rodiča, tak sa v populácii objavilo veľa podobných objektov, ktoré sa líšili len minimálne v dôsledku mutácie a kríženia iných častí objektu ako tej časti, ktorá zabezpečovala stabilitu.

Ako riešenie tohto problému som navrhla postup, ktorým tri najnižšie položené body objektu sú znížené na úroveň najnižšieho z nich, čím sa automaticky stanú opornými bodmi. Toto riešenie zabezpečuje, že ľubovoľný objekt má aspoň tri oporné body a šanca, že bude stabilným objektom, je oveľa vyššia ako keby sa body nezniżovali na úroveň podlahy.

3.4.2 Kritériá pre dobré sedadlá

Prítomnosť sedadla

Po zahrnutí plôch do reprezentácie som sa pokúsila sformulovať kritériá pre plochu vhodnú na sedenie. Prvým kritériom, ktoré som testovala spolu so stabilitou celej stoličky a limitom na rast, bola prítomnosť takmer vodorovnej plochy.

Toto kritérium sa ale ukázalo ako samostatne nie dostatočné, lebo za ideálny útvar považovalo takú stoličku, kde sedadlo sa nachádzalo na zemi, to znamená že takáto stolička bola automaticky aj stabilná, a nemuselo tam byť nič viac, čo by prispievalo k vyššej hodnote fitness funkcie. V extrémnom prípade bola vyvinutá stolička, ktorá sa skladala len z jednej približne vodorovnej plochy.

Výška sedadla nad zemou

Keďže stoličku, ktorá by mala sedadlo na zemi, nepovažujem za celkom ideálne riešenie, pokúsila som sa zahrnúť do fitness funkcie aj podmienku na výšku sedadla nad zemou. Sedadlo je teda dobrým sedadlom len vtedy, ak jeho výška nad zemou je väčšia ako nejaká vopred stanovená konštanta. Fitness funkcia potom ocenila, ak sa v stoličke nachádzalo sedadlo v dostatočnej

výške. Táto úprava fitness funkcie sa však ukázala ako nie celkom vhodná, lebo nie je veľmi pravdepodobné, že v náhodne vygenerovaných útvaroch sa bude nachádzať takéto dobré sedadlo.

Obrázok 3.8: Príklad dobrej stoličky vyvinutej bez použitia kritéria na výšku sedadla.

Počas evolúcie nastal v tomto prípade taký jav, že keď sa náhodou podarilo vyvinúť stoličku s takýmto dobrým sedadlom, tak evolúcia ďalej uprednostňovala len túto stoličku, lebo bola zďaleka najlepšia. Rozmanitosť v populácii prudko klesla, pretože väčšina stoličiek bola nejakou obmenou tej jednej dobrej stoličky. V tomto prípade sa ukázalo, že je veľmi dôležité čiastočne oceňovať aj nie celkom dobré riešenia a tým vlastne evolúciu pohnúť tým správnym smerom k lepším riešeniam. Preto som ohodnocovanie výšky sedadla zmenila na spojitú funkciu, ktorá pomerne oceňuje aj sedadlá, ktoré nemajú dostatočnú výšku. Po tom, čo som do fitness funkcie zahrnula aj čiastočné oceňovanie vodorovných rovín, ktoré ale neboli aj dostatočne vysoko nad zemou, sa problém predčasnej konvergencie k prvému dobrému riešeniu, ktoré sa nájde, vyriešil.

Stabilita sedadla

V priebehu testovania predchádzajúcich kritérií sa objavil aj ďalší problém so sedadlami. Je ním fakt, že stabilná stolička môže mať sedadlo, ktoré je nestabilné. To znamená, že keby sa takéto sedadlo niečím zaťažilo, stolička by sa prevrhla. Tento problém sa dá riešiť viacerými spôsobmi:

- fitness funkcia bude oceňovať, ak sa ťažisko sedadla bude nachádzať v konvexnom obale bodov ležiacich na zemi (výška sedadla nad podlahou sa neberie do úvahy). Toto riešenie som nepovažovala za dobré, lebo je to iba diskrétne kritérium a neviem si dosť dobre predstaviť, ako by sa z neho dalo spraviť spojité kritérium.

- sedadlo by malo dvojnásobnú váhu oproti obyčajnej rovine pri počítaní ťažiska stoličky. Sedadlo by bol dobré, ak by stolička aj po takomto zaťažení sedadla zostala stabilná. Toto riešenie sa mi nezdá celkom dobré z toho dôvodu, že v mojom prístupe abstrahujem od použitého

materiálu a iných vlastností, ktoré ovplyvňujú stabilitu reálnej stoličky. Preto si nemyslím, že je dobré uvažovať o nejakom konkrétnom pomere hmotností a podobne ako prvé navrhované kritérium je to diskrétne kritérium.

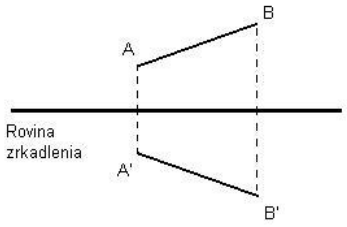
- fitness funkcia bude spojito oceňovať viacej také sedadlá, ktoré majú svoje ťažisko bližšie ku stredu všetkých oporných bodov stoličky (aritmetický priemer príslušných súradníc oporných bodov). Týmto kritériom sú preferované stoličky, ktoré sú stabilné bez ohľadu na to, či na sedadle niekto sedí alebo nie. Výhodou tohto prístupu je, že oceňovanie je možné urobiť spojité v závislosti vzdialenosti ťažiska sedadla od stredu oporných bodov, čo je dôležité pri správnom nasmerovaní evolúcie.

  Pre toto kritérium existuje variant, že vzdialenosť ťažiska sedadla sa nebude počítať od stredu oporných bodov stoličky ale od ťažiska stoličky. Toto riešenie je trochu horšie ako predchádzajúci návrh, lebo ťažisko stoličky sa môže nachádzať blízko pri hranici stability stoličky. V tomto prípade sedadlo, ktorého ťažisko je veľmi blízko ťažiska stoličky, sa môže nachádzať aj mimo konvexného obalu oporných bodov a teda nie je stabilné. Takéto pomerne zlé sedadlo by v tomto prípade malo vysokú fitness za stabilitu, čo nechceme. Tento prípad síce môže nastať aj v pôvodnej verzii tohto kritéria, ale je to oveľa nepravdepodobnejšie a vo väčšine prípadov bude sedadlo naozaj stabilné, ak jeho ťažisko bude dostatočne blízko stredu oporných bodov stoličky.

Obrázok 3.9: Príklady stabilných stoličiek so sedadlom. Sedadlo ale nie je stabilné. (Sedadlo je tmavosivá plocha)

3.4.3 Modifikácie mapovania genotypu na fenotyp

Ľudia prirodzene považujú symetrické veci za krajšie. Stoličky vyvinuté pomocou predchádzajúcich kritérií nie sú symetrické a preto sa nám môžu zdať mierne nevzhľadné, aj keď sú plne funkčné podľa zvolených kritérií. Preto



Obrázok 3.10: Ilustrácia použitia zrkadlovej symetrie (pohľad zhora)

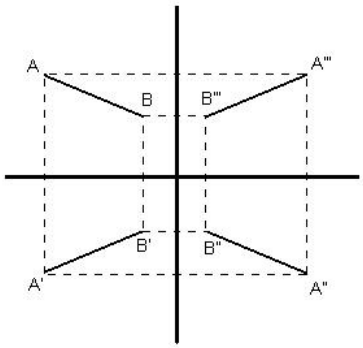
som do svojej práce zahrnula možnosť interpretovať informácie zakódované v chromozómoch viacerými spôsobmi.

Symetria podľa jednej roviny

Najjednoduchšou symetriou je zrkadlová symetria. Používam ju tak, že objekt a jeho obraz v zrkadle považujem za jeden celok. Takto vzniknutý objekt bude súvislý, pretože rovina zrkadlenia prechádza prvým bodom stoličky - začiatkom. Teda minimálne v tomto bode sa budú objekt a jeho zrkadlový obraz dotýkať.

Rovina zrkadlenia je kolmá na rovinu podlahy. V tomto prípade sa oporné body zobrazia v zrkadle na body, ktoré budú tiež opornými bodmi. Preto stačí pri prevádzaní genotypu na fenotyp znižovať na úroveň najnižšieho bodu ešte druhý najnižší bod a objekt bude mať aspoň štyri oporné body. Informácie zachytené v genotype sú spracované tak, že pre každý bod vypočítaný pri klasickom prevode genotypu na fenotyp sa vypočíta aj k nemu zrkadlovo symetrický bod a zahrnie sa do stoličky. Štruktúry ako čiary a plochy sú zachované.

Ak v tomto prípade použijeme kritérium stability sedadla, ktoré oceňuje blízkosť ťažiska sedadla ku stredu oporných bodov, bude to mať zaujímavé dôsledky. Lepšie budú tie stoličky, ktoré budú mať sedadlá v strede stoličky (podľa roviny zrkadlenia), pretože v tomto prípade sa ťažisko aj stred oporných bodov nachádzajú v rovine zrkadlenia. Teda budú preferované stoličky, ktoré síce majú dve sedadlá, ale tieto sedadlá sa prekrývajú a tvoria tak akoby len jedno sedadlo v strede. Prípad, keď stolička má dve sedadlá vzdialené od seba viacej, hodnotí fitness funkcia nižšie.



Obrázok 3.11: Príklad použitia dvojnásobnej symetrie (pohľad zhora)

Symetria podľa dvoch kolmých rovín

Podobným spôsobom ako v predchádzajúcom prípade je robená aj symetria podľa dvoch rovín. V tomto prípade sú však ku každému bodu dopočítané ďalšie tri prislúchajúce body. Priesečník rovín zrkadlenia sa nachádza v začiatočnom bode, preto sú aj tieto objekty súvislé.

Výhodou tejto modifikácie je, že ťažisko takéhoto objektu sa nachádza vždy v priesečníku rovín zrkadlenia, teda vlastne v strede objektu. Okrem prípadu, ked najnižší bod sa nachádza presne v strede objektu alebo v niektorej z rovín zrkadlenia, je takýto útvar automatický stabilný. Dôvodom je, že najnižší bod objektu sa trikrát zduplikuje a objekt ma aspoň štyri oporné body rozmiestnené súmerne okolo ťažiska objektu.

Pridanie dosky stola

Takéto dvojnásobne symetrické útvary mali väčšinou štyri nohy, čím pripomínali nohy stola. Chýbala už len doska položená navrchu. Preto som pridala možnosť pridania dosky stola takým spôsobom, že bude položená na štyroch najvyšších bodoch objektu. Veľkosť dosky je volená tak, aby pokrývala práve celý útvar pri pohľade zvrchu.

Takto definovaná doska môže mať jeden problém. Ak sa najvyšší bod nachádza v priesečníku rovín, tak je to vlastne len jeden bod a na takomto bode by doska spčívala asi len veľmi labilne. Preto som navrhla kritérium pre dosku stola, ktoré oceňuje prípad, ak je pomer plochy vymedzenej opornými bodmi dosky a plochy celej dosky väčší, ako nejaká konštanta.

Pridala som aj podobné kritérium pre rozstup nôh stola, pretože kritérium stability tak, ako bolo definované pre stoličky, nemá v prípade stolov význam. Toto z kritérium oceňuje väčší pomer plochy ohraničenej opornými bodmi

útvaru a plochy dosky stola. Použitie tohoto kritéria vedie k tomu, že za lepšie sú považované také stoly, ktoré sú stabilnejšie.

Tieto kritériá sa však ukázali ako pomerne slabé, pretože väčšina náhodne vygenerovaných stolov ich vo veľkej miere spĺňa. Preto nie je veľký tlak pri evolúcii zlepšovať riešenia.

Kapitola 4

Výsledky

4.1 Úspešnosť kritérií

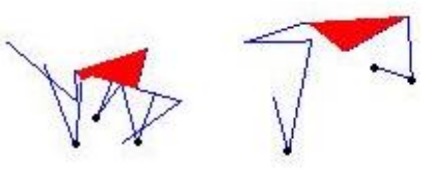
V nasledujúcej časti by som chcela zhrnúť, aké výsledky som dosiahla pri použití kritérií, ktoré som popísala v predchádzajúcej kapitole.

4.1.1 Obmedzenia na počet plôch a čiar

V každom prípade odporúčam použiť pri evolúcii také nastavenie obmedzujúcich kritérií na počet čiar, aby boli aspoň minimálne postihy za prekročenie nejakej hornej hranice. Ukázalo sa totiž, že objekty majú tendenciu rásť do veľkých rozmerov napriek tomu, že s rastúcou veľkosťou sa nezväčšuje pravdepodobnosť, že kritériá budú ľahšie splnené.

Pre menšiu stoličku je nižšia pravdepodobnosť toho, že sa do nej zmestí dostatočné množstvo komponentov, ktoré splnia požadované kritériá. Ak je stolička malá, je menšia pravdepodobnosť, že sa tam bude nachádzať celá časť jedného z rodičov, ktorá u rodiča spôsobuje, že je splnené niektoré z kritérií. Napríklad najmenšia stolička, ktorá je stabilná a má sedadlo v dostatočnej výške, musí obsahovať aspoň štyri úsečky a jednu plochu. Preto by mohli mať väčšie stoličky výhodu pred menšími, aj keď priamo to na ich schopnosť spĺňať kritériá nevplýva. Tento argument funguje len do istej veľkosti objektov. Po prekročení nejakej hranice na veľkosť je už v každom objekte dostatočné množstvo komponentov na splnenie kritérií. Ďalšie komponenty by nemali veľmi zvyšovať pravdepodobnosť splnenia kritérií vo väčších stoličkách.

Zaujímavý je aj fakt, že do veľkých rozmerov rastú pri evolúcii aj stoly. V tomto prípade totiž počet komponentov ovplyvňuje kvalitu riešenie len minimálne a aj to len v prípade, ak je počet komponentov príliš malý. To znamená, že počet komponentov prakticky nemá vplyv na kvalitu riešenia.



Obrázok 4.1: Príklad vyvinutých stoličiek (čierne body označujú body na zemi)

Napriek tomu objekty stále rastú. Tento jav je pravdepodobne spôsobený tým, že ak pri krížení dvoch rodičov vznikne jedna malá a jedna veľká stolička, tak malá stolička bude mať pravdepodobne nižšiu fitness. Z obidvoch potomkov bude teda úspešnejší ten väčší, čím sa priemerná veľkosť stoličiek v populácii zväčšuje.

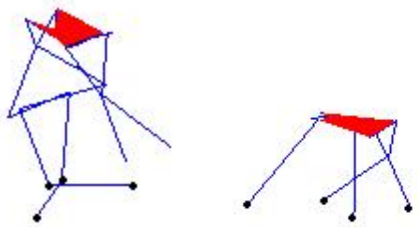
Keďže plochy sa môžu nachádzať len na už existujúcich čiarach, nie je nutné zhora obmedzovať počet plôch, ak na tom nemáme nejaký konkrétny záujem. Kritérium obmedzujúce počet plôch je vhodné použiť ak napríklad chceme, aby v stoličke alebo stole neboli žiadne plochy alebo len nejaký malý počet plôch, alebo naopak ak chceme zaistiť, aby tam nejaké plochy určite boli.

Pri použití kritérií na počet čiar a plôch určuje postih za prekročenie zadaného počtu, akú veľkú toleranciu dávame pri vyhodnocovaní tohto kritéria prekročeniu zadaného limitu. Ak je postih malý, tak sa malé prekročenie limitu hodnotí len málo záporne, ak je postih vysoký, tak aj jedna úsečka alebo plocha navyše spôsobí radikálne zníženie hodnoty fitness celého objektu.

4.1.2 Kritérium stability

Stabilita je nutná podmienka pre každý objekt, ktorý ma byť aspoň trochu dobrý a použiteľný. Je to ale aj najproblematickejšie kritérium, ktoré používam pri evolúcii stoličiek.

Problém je v tom, že pri takom krížení, ako som ho zadefinovala, nie je dostatočne veľká pravdepodobnosť, že potomkovia budú stabilní, ak mali stabilných rodičov. Kríženie môže totiž nastať v mieste, ktoré spája oporné body a tým sa odoberie časť stoličky, ktorá zabezpečovala stabilitu. Je malá pravdepodobnosť, že iná časť, ktorá sa na miesto kríženia pridá z druhej stoličky bude dobrou náhradou odobratej časti a bude obsahovať oporný bod, ktorý zabezpečí stabilitu celej stoličky. Tento problém čiastočne riešim pomocou znižovania bodov na úroveň najnižšieho bodu v stoličke. Po tejto



Obrázok 4.2: Príklad vyvinutých zrkadlovo symetrických stoličiek (čierne body označujú body na zemi)

úprave toto kritérium nie je bezproblémové, ale má svoje opodstatnenie.

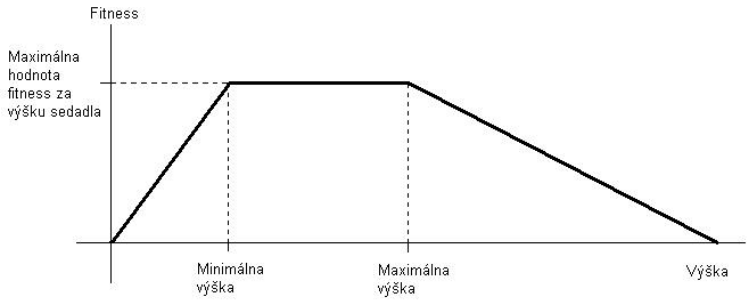
4.1.3 Kritériá pre stoličku

Dobré výsledky pri evolúcii stoličiek sa podarilo dosiahnuť s nasledovným nastavením:

Stabilita	10
Rovnosť sedadla	10
Výška v zadanom intervale	10
Stabilita sedadla	10
Minimálna výška	2
Maximálna výška	5

Tabuľka 4.1: Nastavenie fitness funkcie pre evolúciu stoličiek

Rovnosť sedadla oceňuje prítomnosť aspoň jednej približne vodorovnej plochy v objekte. Nasledujúce dve kritériá sa vyhodnocujú tak, že každé kritérium sa vyhodnotí pre všetky potenciálne sedadlá (približne vodorovné plochy), ktoré sa nachádzajú v stoličke. Za výslednú hodnotu sa zoberie maximálna dosiahnutá hodnota. Kritérium výšky maximálne oceňuje, ak sa v stoličke nachádza sedadlo, ktorého výška je v zadanom intervale. Ak sa v stoličke nachádza sedadlo, ktorého výška sa nenachádza v uvedenom intervale, tak jeho fitness je vypočítaná pomerom jeho reálnej výšky k minimálnej alebo maximálnej zadanej výške, podľa toho, či sa nachádza pod alebo nad intervalom. To znamená, že pri nastavení uvedenom v tabuľke 4.1 sedadlo, ktoré sa bude nachádzať vo výške 1, získa 5 bodov za výšku sedadla. Podobným spôsobom sa vypočítava aj hodnota fitness za stabilitu sedadla, pričom



Obrázok 4.3: Priebeh funkcie použitej na vyhodnocovanie fitness za výšku sedadla.

maximálna hodnota je zadaná používateľom a reálna hodnota fitness sedadla klesá lineárne so vzďaľovaním sa ťažiska sedadla od ideálnej pozície.

Pri takomto nastavení nie je veľmi dôležitá hodnota, ktorá je použitá na ocenenie splnenia niektorého z kritérií, pokiaľ sú všetky tieto hodnoty pomerne vyvážené.

4.1.4 Kritériá pre stôl

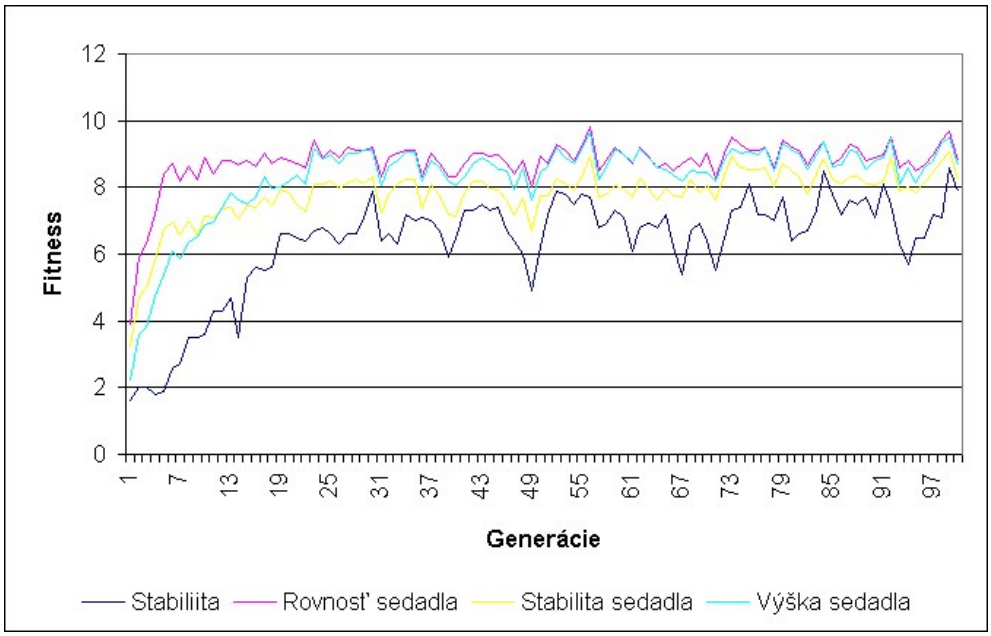
Pri evolúcii stolov nemá zmysel používať žiadne z kritérií pre sedadlá.

Evolúcia stolov pri takom nastavení fitness funkcie, ktoré oceňuje stabilitu objektu, dostatočný rozstup nôh stola a oporných bodov dosky stola nevedie k vývinu lepších stolov. Problémom je, že pre takto definované kritériá je väčšina náhodne vygenerovaných stolov dobrá a teda už tu nie je dostatočný priestor na zlepšovanie riešení. Použitie kritéria pre stabilitu objektu je v prípade, že sa používa kritérium pre dostatočný rozstup nôh stola zbytočné, pretože obidve tieto kritéria oceňujú tú istú vec, len jedno je diskrétne a druhé spojité.

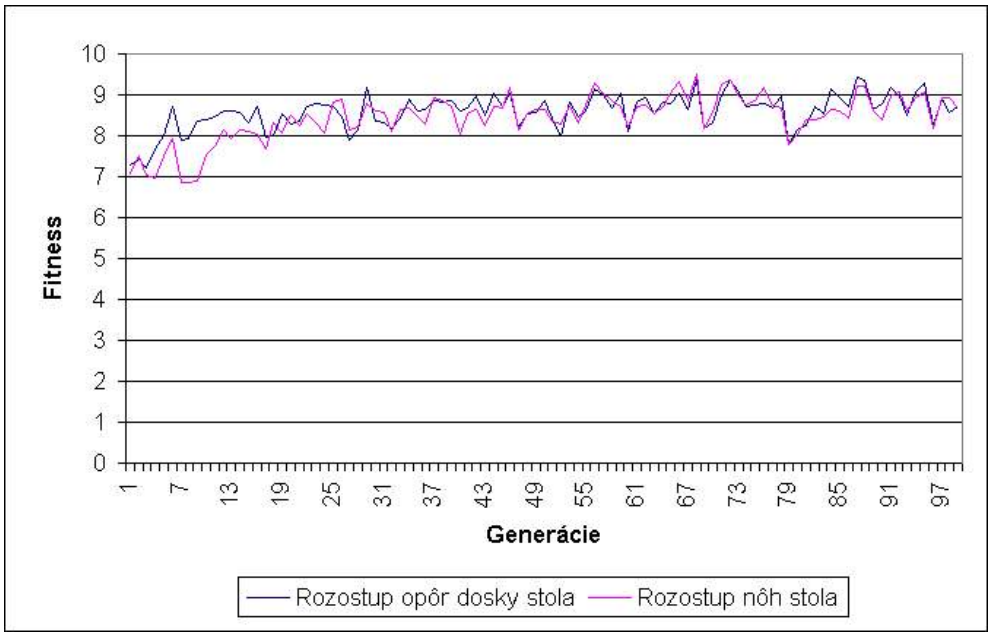
Evolúcia sa však v tomto prípade dá použiť na vývin stolov, ktoré sú niečím špecifické. Napríklad sa dajú pomocou použitia kritérií na veľkosť vyvinúť jednoduché stoly, zložité stoly, stoly bez rovín a podobne.

4.2 Záver

Vývoj stoličiek podľa mnou zvolených kritérií je úspešný. Nepodarilo sa mi však sformulovať nejaké kritérium pre operadlo, čím sa dosť zmenšila množina stoličiek, ktoré sa dajú vyvinúť, pretože sa dajú vyvinúť len stoličky bez operadiel. Na obrázku 4.4 je možné vidieť priebeh priemerných hodnôt



Obrázok 4.4: Vývoj priemerných hodnôt fitness pri evolúcii stoličiek.



Obrázok 4.5: Vývoj priemerných hodnôt zložiek fitness funkcie pri evolúcii stola (maximálna hodnota fitness pre obidve kritériá je 10)

fitness za jednotlivé kritériá.

Kritérium pre stabilitu dosahuje najmenšiu priemernú hodnotu zo sledovaných kritérií, ale aj tak sa jeho hodnota nachádza nad polovičnou hodnotou. Teda približne od dvadsiatej generácie je viac ako polovica stoličiek v každej populácii počas evolúcie stabilná. Problémom stability objektu je, že na splnení tohoto kritéria podieľa viacej komponentov stoličky. Je teda vyššia pravdepodobnosť, že pri krížení sa poruší štruktúra, ktorá stabilitu zabezpečuje. Nejaká jednoduchá zmena v krížení alebo reprezentácii, ktorá by tento problém riešila, podľa mňa neexistuje.

Rovnosť sedadla a výška sedadla sa držia blízko pri maximálnych možných hodnotách. Príčinou tohto javu je, že na splnenie týchto kritérií stačí, aby bola prítomná jedna vhodná rovina. Ostatné komponenty, ktoré sa v stoličke ešte nachádzajú, ovplyvnia tieto kritériá len minimálne.

Stabilita sedadla dosahuje v priemere trochu nižšie hodnoty. Tento fakt je spôsobený tým, že dosiahnuť maximálnu hodnotu fitness pre toto kritérium je takmer nemožné. Je veľmi nepravdepodobné, že sa ťažisko nejakého sedadla bude nachádzať presne nad stredom oporných bodov. Preto sú hodnoty fitness za toto kritérium nižšie, ale napriek tomu považujem tieto hodnoty za dobré.

4.2.1 Porovnanie reprezentácií

Základným rozdielom medzi mojou reprezentáciou a reprezentáciami použitými v [2,3] je v tom, že ja používam ako stavebné prvky objekty, ktoré nie sú priestorové, nemajú objem. V obidvoch príkladoch iných reprezentácií používajú ako základné stavebné prvky priestorové telesá. Rozdiel je v tom, že ak majú len jedno teleso, je automaticky stabilné. Ak mám ja len jednu čiaru, tak je stabilná len vtedy, ak je položená na zemi, inak je nestabilná. Teda v mojom prípade je oveľa ťažšie dosiahnuť stabilitu. Podobne má moja reprezentácia problém so stabilitou pri krížení.

Na druhej strane je moja reprezentácia oveľa flexibilnejšia, čo sa týka množstva útvarov, ktoré sú reprezentovateľné. Napríklad v reprezentácii pomocou kociek je problém urobiť nejaký šikmý útvar, čiaru. Na nasimulovanie takéhoto javu je potrebných veľa kociek. V mojej reprezentácii je to len jeden komponent. Tento problém sa dá riešiť aj pomocou iných reprezentácií, ale len za cenu toho, že kocky, ktoré používajú, by boli menšie vzhľadom na vyvíjané útvary. Toto zmenšenie kociek by pravdepodobne zvýšilo výpočtovú náročnosť evolúcie.

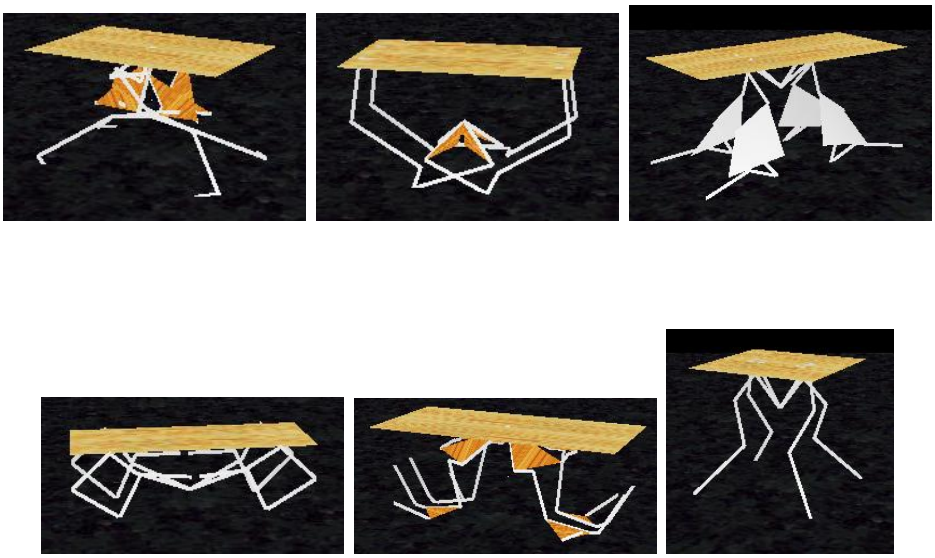
Obrázok 4.6: Príklad minimalistických stolov bez plôch, ktoré sa dajú vyvinúť.

4.3 Čo by sa dalo ešte robiť

Na príklade stola vidno, že pri vhodnej interpretácii ľubovoľných dát, je možné vo väčšine prípadov vyrobiť už dobrý stôl. Podobným spôsobom by sa dali vyrábať určite aj iné objekty, ak vieme presne popísať, čo od nich očakávame. Bolo by teda zaujímavé uvažovať nad tým, akým spôsobom interpretovať náhodné dáta tak, aby z toho vždy vznikla už pomerne dobrá stolička.

V niečom podobnom vidím druhý prístup, akým možno poňať túto tému, t.j. vyrábať už dobré útvary a evolúciu použiť na splnenie nejakých špecifickejších (konkrétnejších) kritérií. Pri takomto prístupe by bolo potrebné nejakým spôsobom umožniť nastavovať jemnejšie detaily, pomocou ktorých by bolo možné presne vyšpecifikovať aj väčšie detaily želaného dizajnu stoličky. Evolúcia by potom vyberala z už dobrých stoličiek len tie lepšie podľa zvolených estetických kritérií.

Na riešenie takéhoto problému by bola vhodná aj moja reprezentácia, pretože problém s krížením by tu nenastával. Dôvodom je, že ak sa ľubovoľné dáta dajú previesť na dobrú stoličku, tak by kríženie nemohlo vyrobiť tak zlé dáta, že by z toho bola zlá stolička. Podobne ako pri krížení stolov v mojom prístupe nevznikajú stoly, ktoré by boli horšie ako náhodné stoly. Vidieť to na grafe na obrázku 4.5, kde hodnoty oboch kritérií sa držia počas evolúcie na vysokých hodnotách.



Obrázok 4.7: Príklad náhodných stolov

Dodatok A

Užívateľská príručka

A.1 Všoebecný popis

Tento program je určený na evolúciu nábytku, konkrétne stoličiek a stolov. Objekty vyvinuté pomocou tohoto programu majú slúžiť na inšpiráciu dizajnéra pri navrhovaní nábytku.

Program pri spustení vytvorí náhodnú populáciu objektov podľa nastavení, ktoré sú uložené v ini-súbore. Potom je možné na tejto populácii spustiť evolučný algoritmus. Fitness funkcia, ktorá sa pri tejto evolúcii vyhodnocuje, sa dá v programe nastaviť. Podľa týchto nastavení sú vyvíjané potom objekty.

Po skončení zadaného počtu generácií je možné si postupne prezrieť všetky objekty v súčasnej generácií. Neskôr sa dajú spustiť ešte ďalšie generácie. Pri prezeraní je možné si prispôsobiť uhol pohľadu a vzdialenosť.

Objekty, ktoré vyhovujú našim predstavám, je možné uložiť do súboru. Takto uložené objekty je neskôr možné pomocou tohoto programu zase otvoriť. Pri načítaní objektu zo súboru sa zruší aktuálna populácia.

Podrobnosti sú popísané v nasledujúcich kapitolách.

A.2 Štruktúra menu

- Stolička

  -> Ulož - uloží aktuálnu stoličku do súboru na disku.

  -> Otvor - otvorí stoličku uloženú v súbore, aktuálna populácia je zrušená.

  -> Podrobnosti - zobrazí podrobnosti o aktuálnej stoličke.

- Populácia
  - -> Nová populácia - vytvorí novú populáciu stoličiek s aktuálnymi nastaveniami. Novú populáciu je nutné vytvoriť vždy, keď sa zmenili nastavenia, aby sa tieto nové nastavenia prejavili.
  - -> Nastavenie - nastavovanie vlastností stoličiek, aké majú byť v populácii. Aby sa zmeny prejavili, je nutné urobiť novú populáciu.
- Fitness
  - -> Nastavenia - nastavovanie vlastností fitness funkcie.
  - -> Podrobnosti - zobrazí podrobné informácie o štruktúre a hodnotách fitness funkcie pre aktuálnu stoličku.
- Koniec - ukončí program.

A.3 Hlavné okno programu

V hlavnom okne programu je zobrazený aktuálny objekt. So zobrazeným objektom je možné manipulovať nasledujúcimi spôsobmi: zmenšenie (oddialenie) a zväčšenie (priblíženie) objektu, posunutie pohľadu smerom nahor a nadol. Tlačidlá `Zoom+` a `Zoom-` slúžia na priblíženie a oddialenie zobrazovaného objektu. Podobne tlačidlá `Rotuj+` a `Rotuj-` slúžia na zmenu výšky pohľadu. Pomocou tlačidiel `<--` a `-->` je možné postupne prezerať všetky objekty v populácii. Číslo medzi šípkami určuje poradové číslo objektu v populácii. Tlačidlo `Na začiatok` slúži na presun na prvý objekt v populácii.

Fitness udáva hodnotu fitness funkcie pre aktuálny objekt, generácia je hodnota aktuálnej generácie.

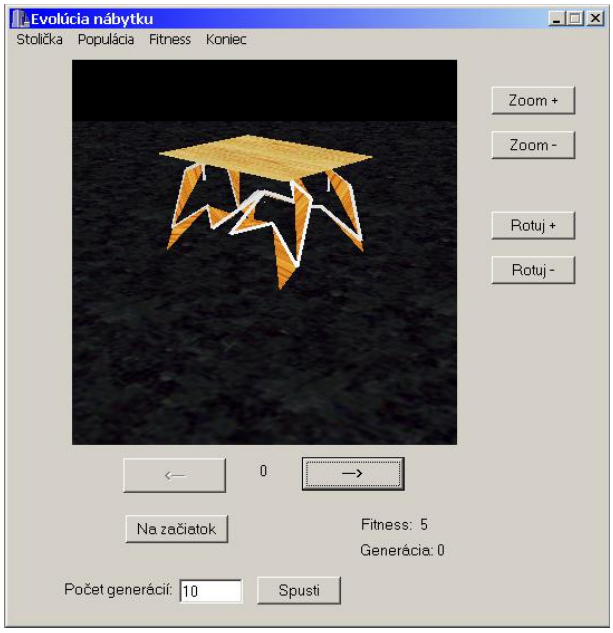
Tlačidlo `Spusti` slúži na spustenie evolúcie. Počet generácii, ktoré sa majú vykonať, sa zadáva do políčka `Počet generácií`

A.4 Nastavovanie parametrov programu

Program umožňuje rôzne nastavenia vlastností objektov v populácii a vlastností fitness funkcie. Deje sa to prostredníctvom dvoch formulárov, ktoré teraz detailne popíšem.

A.4.1 Nastavenie fitness funkcie

Tento formulár slúži na nastavenie vlastností fitness funkcie, ktorá sa bude vyhodnocovať pri evolúcii.



Obrázok A.1: Hlavná obrazovka programu

Ak chceme, aby sa niektoré kritérium bralo do úvahy, je potrebné zaškrtnúť príslušné políčko na formulári. Ak sa pri kritériu dajú nastavovať ďalšie podrobnosti, pri zaškrtnutí sa nám sprístupní možnosť editovať hodnoty jednotlivých položiek podrobnejšieho nastavenia.

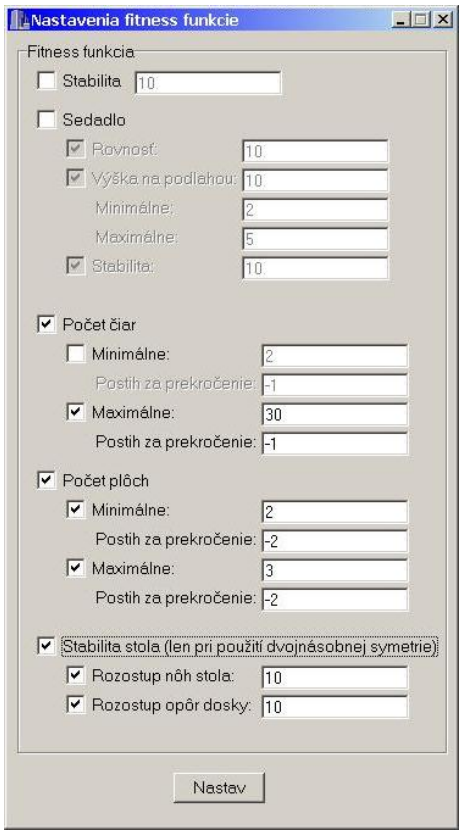
Stabilita je kritérium, ktoré oceňuje stabilitu objektu. Hodnota fitness, ktorú získa stabilný objekt, je špecifikovaná v príslušnom políčku.

Sedadlo zahŕňa všetky kritériá pre sedadlá.

- *Rovnosť* oceňuje prítomnosť približne vodorovnej plochy v objekte.

- *Výška* oceňuje, ak v objekte existuje sedadlo, ktoré sa nachádza v intervale zadanom v políčkach `Minimálne` a `Maximálne`. Odporúčaná maximálna hodnota v týchto dvoch políčkach je 10.

- *Stabilita* oceňuje stabilitu sedadla, t.j. blízkosť ťažiska sedadla k aritmetickému stredu oporných bodov objektu. Maximálna hodnota, ktorú môže objekt získať za toto kritérium je špecifikovaná v príslušnom políčku.



Obrázok A.2: Formulár na nastavovanie fitness funkcie

Počet čiar slúži na obmedzenie požadovaného minimálneho alebo maximálneho počtu čiar, z ktorých sa skladá objekt.

Nie je nutné vždy použiť obidve obmedzenia. Maximálny požadovaný počet čiar by mal byť prirodzene väčší ako minimálny požadovaný počet čiar. Takéto nastavenie nie je neprípustné, ale pozitívny efekt takto nastaveného kritéria bude nulový.

V políčku `Minimálne` sa špecifikuje minimálny požadovaný počet čiar. Do políčka `Postih za prekročenie` sa zadáva záporné číslo. O takúto hodnotu bude znížená výsledná hodnota fitness za každú čiaru, ktorá chýba k dosiahnutiu zadaného limitu.

Podobne sa nastavuje limit a postih za prekročenie aj pre maximálny počet čiar. Postih sa uplatňuje pre každú čiaru, ktorá presahuje zadaný limit.

Počet plôch slúži na obmedzenie počtu plôch v objekte. Nastavuje sa tak isto ako kritériá pre počet čiar.

Počet plôch je vždy menší alebo rovný ako počet čiar, pretože plocha sa musí vždy nachádzať na nejakej čiare a na každej čiare môže byť najviac jedna plocha. Preto nemá zmysel mať väčší horný limit na počet plôch ako na počet čiar. Takéto nastavenie je však korektné a nespôsobí neštandardné správanie programu.

Stabilita stola je kritérium, ktoré sa používa pri evolúcii stolov, t.j. dvojnásobne symetrických útvarov s doskou. V ostatných prípadoch sa toto kritérium neberie do úvahy, aj keby bolo označené, pretože to nemá zmysel.

V políčku `Rozstup nôh stola` je zadaná maximálna hodnota, ktorá sa dá dosiahnuť, ak je rozstup nôh stola dostatočný. Dostatočný je vtedy, ak plocha, ktorá je určená opornými bodmi, má aspoň polovičnú plochu vzhľadom na dosku stola. Ak je plocha menšia, tak sa berie pomerná časť z maximálnej možnej hodnoty podľa plochy určenej opornými bodmi. ???

Hodnota v `Rozstup opôr dosky` je vyhodnocovaná podobne ako predchádzajúce kritérium. Berie sa však do úvahy pomer plochy určenej opornými bodmi dosky (body, na ktorých je doska položená) a plochy celej dosky stola. Maximálnu hodnotu stôl získa, ak je plocha ohraničená opornými bodmi dosky väčšia ako devätina plochy dosky stola.

Kliknutím na tlačidlo `Nastav` sa uložia hodnoty nastavení. Nové nastavenie sa prejaví pri najbližšom vytváraní novej populácie.

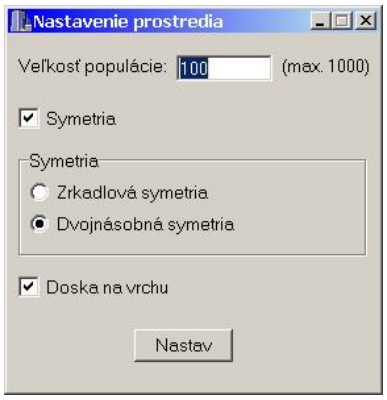
Vo formulári sú akceptované len celé čísla. Pri pokuse o vyplnenie políčok reťazcami alebo desatinnými čislami, program odmietne prijať takéto údaje a zobrazí upozornenie o nevhodnom formáte zadávaných údajov.

A.4.2 Nastavenie vlastností objektov v populácii

Tento formulár umožňuje nastavovanie vlastností objektov v populácii ako sú symetria a prítomnosť dosky stola. Tiež sa tu nastavuje počet objektov, ktoré majú tvoriť populáciu evolučného algoritmu.

Veľkosť populácie umožňuje nastaviť počet objektov, ktoré majú byť v populácii. Maximálna prípustná hodnota je 1000 objektov.

Symetria aktivuje používanie symetrie v objektoch. Druh symetrie, aký sa má použiť sa vyberá pomocou prepínačov s hodnotami `Zrkadlová symetria` a `Dvojnásobná symetria`



Obrázok A.3: Formulár na nastavovanie vlastností objektov v populácii

Doska na vrchu umožňuje nastaviť, či sa má na objekty v populácii klásť doska stola. Rozlišujeme tým, či objekty v populácii sú stoly alebo stoličky.

Kliknutím na tlačidlo `Nastav` sa nastavenia uložia. Nové nastavenie populácie sa prejaví pri vytváraní novej populácie.

A.5 Zobrazenie podrobných údajov

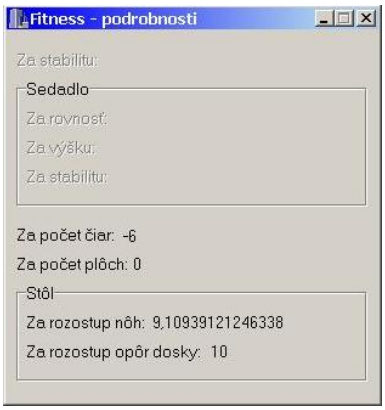
Program umožňuje zobrazenie detailných údajov o objekte a o hodnotách a štruktúre fitness funkcie, ktorá sa počíta.

A.5.1 Podrobnosti fitness funkcie

Na tomto formulári môžeme vidieť detailný rozpis hodnôt fitness funkcie podľa jednotlivých kritérií. Je tu zobrazená hodnota fitness pre každé kritérium, ktoré sa berie do úvahy pri výpočte fitness funkcie. Hodnota fitness funkcie je súčtom hodnôt fitness za jednotlivé kritériá, ktoré sa berú do úvahy. Kritériá, ktoré sa nevyhodnocujú sú na formulári zobrazené ako zašednuté.

A.5.2 Podrobné údaje o objekte

Na tomto formulári môže vidieť informáciu o tom, z akých komponentov sa objekt skladá. Je tu informácia o počte čiar a plôch, o celkovej hodnote fitness funkcie. Môžeme tu tiež vidieť informáciu o prítomnosti aspoň jedného sedadla a o stabilite objektu.



Obrázok A.4: Podrobné popísanie hodnôt fitness funkcie

Obrázok A.5: Podrobné popis vlastností objektu

A.6 Príklady nastavení pre rôzne ciele evolúcie

Teraz uvediem príklady nastavení fitness funkcie pre rôzne ciele evolúcie. V každom prípade odporúčam použiť obmedzenie na maximálny počet čiar v objekte. Ak sa toto kritérium neberie do úvahy, veľkosť objektov môže presiahnuť maximálnu povolenú veľkosť a vtedy je program ukončený.

V tabuľke A.1 je uvedené základné nastavenie pre evolúciu stoličiek. Ďalšie modifikácie fitness funkcie sa dajú spraviť pomocou určenia obmedzení na počet plôch, zmeny limitu na maximálny počet čiar a určenia minimálneho limitu na počet čiar.

V tabuľke A.2 je príklad základného nastavenia na evolúciu stolov. Podobne ako pri evolúcii stoličky sa dajú ďalšie modifikácie fitness funkcie robiť pomocou kritérií na počet čiar a plôch. Pri evolúcií stolov je dobré zvýšiť limit na maximálny počet čiar, pretože v takýchto objektoch je čiar viac.

Stabilita	10
Sedadlo:	
Rovnosť	10
Výška nad podlahou	10
Minimálna výška	2
Maximálna výška	5
Stabilita	10
Počet čiar:	
Maximálne	20
Postih za prekročenie	-1

Tabuľka A.1: Príklad nastavenia fitness funkcie pre evolúciu stoličiek

Počet čiar:	
Maximálne	40
Stabilita stola:	
Rozstup nôh	10
Rozstup opôr dosky	10

Tabuľka A.2: Príklad nastavenia fitness funkcie pri evolúcii stolov

A.7 Možné problémy

Program by mal bez problémov fungovať na väčšine počítačov s novšími grafickými kartami. Osobne som ho testovala aj na starších počítačoch a fungoval bez problémov, len možno trochu pomalšie :).

Ak napriek tomu pri štarte programu nastane chyba "Nepodarilo sa vytvoriť Rendering Context!", pravdepodobne ovládač grafickej karty nepodporuje grafickú knižnicu OpenGL. Odporúčam stiahnutie si najnovšieho ovládača grafickej karty, ktorý podporuje OpenGL, zo stránky výrobcu grafickej karty.

Teraz uvediem niektoré oznamy o chybách, ktoré sa môžu vyskytnúť počas práce s programom:

Chybový oznam na obrázku A.6 sa objaví pri pokuse zadať hodnotu, ktorá nie je prípustná. Najčastejšie je to zadanie reťazca alebo desatinného čísla do políčka, kam má ísť celé číslo.

Chybový oznam na obrázku A.7 sa objaví, ak počas evolúcie niektorý objekt presiahol povolenú hranicu na veľkosť objektov. Program po zobrazení tohoto chybového hlásenia skončí.

Obrázok A.6: Chyba 1

Obrázok A.7: Chyba 2

Riešením je obmedziť striktnejšie rast objektov počas evolúcie pomocou zníženia limitu na počet čiar alebo pomocou prísnejšieho postihu za prekročenie tohoto limitu.

Literatúra

[1] P. J. Bentley (Ed.): *Evolutionary design by computers*, Morgan Kaufmann, 1999

[2] P. J. Funes, J. B. Pollack: *Computer Evolution of Buildable Objects*, eds. P. Husbands and I. Harvey, Fourth European Conf. on Artificial Life, MIT Press, strany 358–367, 1997, http://www.demo.cs.brandeis.edu/papers/edc98.pdf

[3] G. S. Hornby, J. B. Pollack: *The Advantage of Generative Grammatical Encoding for Physical Design*, Proceedings of the 2001 Congress on Evolutionary Computation CEC2001, strany 600–607, IEEE Press, 2001, http://ic.arc.nasa.gov/people/hornby/papers/hornby_cec01.pdf

[4] M. Schein: *Applied Generative Procedures in Furniture Design*, ed. Celestino Soddu, Proceedings of 5th International Conference Generative Art GA2002, Milan, 2002, http://www.generativeart.com/papersGA2002/21.pdf

[5] A. G. De Silva Garza, M. L. Maher: *Evolving Design Layout Cases to Satisfy Feng Shui Constraints*, eds. J. Gu and S. Wei, Proceedings of the Fourth Conference on Computer-Aided Architectural Design Research in Asia, Shanghai, China, 1999, www.arch.usyd.edu.au/ chris_a/MaherPubs/CAADRIA99.pdf

[6] B. Colakoglu: *An Informal Shape Grammars for Interpolations of Traditional Bosnian Hayat Houses in a Contemporary Context*, ed. Celestino Soddu, Proceedings of 5th International Conference Generative Art GA2002, 2002, http://www.generativeart.com/papersGA2002/15.pdf

[7] V. Kvasnička, J. Pospíchal, P. Tiňo: *Evolučné algoritmy*, Vydavateľstvo STU, 2000