**Relation Grapher**

Autoři: Filip Špimr a Michal Špimr

Relation Grapher je program na vykreslování relací (také průnik a sjednocení relací) + umí vykreslovat i funkce, ale jak je vše počítáno přes pixely, tak jsou funkce většinou tečkované (například u tan(x) je vidět tento fenomén velmi dobře). Program ale má druhý algoritmus (dá se zapnout checkboxem „Better functions“) pro počítání funkcí, který bohužel funguje jen na spojité funkce, u nespojitých funkcí jsou grafy matematicky nekorektní (asymptoty jsou také vykresleny jako část funkce, protože kvůli nepřesnosti výpočtů hodnoty blížící se k nevlastnímu bodu jsou považovány jako korektní – což není dobře).

 Program vychází z nedostatků Geogebry, která (jelikož je graficky vektorová) neumí dobře vykreslovat relace. Největší přínos programu je podle mě možnost používat oboje hodnoty X a Y v jedné funkci - např. tan(x\*y) < y. Normální člověk by toto lehko numericky nevypočítal. Mimo použití programu na jednoduché relace jde také program použít na vykreslování umělecky kvalitních obrázků (různými kombinacemi funkcí, X a Y jde dosáhnout zvláštních vzorů, někdy až fraktálně vypadajících), které se potom dají uložit do .png formátu.

 Vstupní relace mohou mít v sobě proměnné ( **x** **y** ), znamínka ( **+ - / \* ^**), závorky () a [] – hranatými závorkami se uzavírají relace, mezi relacemi se může použít **and** (průnik) nebo **or** (sjednocení), poté funkce – **sin**, **cos**, **tan**, **ln**, **log**, **sqrt** (druhá odmocnina), **sgn -** (nedělá dobré funkce – graf je přebarven osami, ale na relace se použít dá), **abs** (absolutní hodnota),  **round** (zaokrouhlení, v matematice se moc nepoužívá, není to celá část čísla!), **trunc** (nedává ohled na desetiná čísla, také v matematice nepoužívané). Také jsou nadefinované konstanty **PI** ($π$) a **e** (eulerovo číslo). Co se týče operátorů, tak jsou nadefinované **=**, **<>** (nerovná se), **<**, **>**, **<=**, **>=**.

 Jak se program používá je ukázáno v obrázku „Relation\_Grapher\_Navod.png“, který je ve složce programu.

 Knihovny, které používá program jsou: **relation.grapher.program** (obsahuje grafickou část), **eval\_unit, eval\_operation, eval\_algorithm, eval\_context, eval\_stack** (těchto 5 knihoven je použito na matematickou část programu), **BGRABitmap –** složka (bitmap – používám ke kreslení, je to mnohem rychlejší než přímo kreslit na form.canvas)

**Hlavní algoritmy**

Grafická část programu:

* **drawExpression(myForm)** – na začátku programu se vyrobí bitmapa, která má velikost formu. V této procedůře se posílají reálné hodnoty x a y převedené z pixlových hodnot form.canvas (začíná se od levého horního rohu formu, kde jsou hodnoty [0,0]). Ve dvou for cyklech se jde přes každý pixel a hodnota jeho reálných souřadnic je poslána přes **evalBoolExpression(defx,defy,pixelsize) –** v této procedůře se vypočítá jestli pro dané hodnoty dává daný relační výraz smysl – podle toho vyhodí true nebo false, pokuď to platí, tak se daný pixel v bitmapě vybarví na oranžovo. Když procedůra skončí tak se bitmapa vybarví na form.canvas – přes to se ještě jednou vybarví osy a graf je hotov.
* **TGrapherForm.ExpressionChange() –** tato metoda se stará o to, jestli je napsaný výraz v pořádku, pokaždé kdy se změní, se výraz pošle přes **compileBoolExpression()**, tam se rozparsuje a pokuď se najde chyba v syntaxi, tak vrací exception, která je napsána v label pod editem, kam se píšou výrazy. V lablu je napsáno, kde a jaká je chyba ve výrazu. Pokuď je relační výraz správně, je použit **drawExpressionMinecraft()**, který vykreslí „kostičkovanou“ verzi grafu, toto je používáno pro rychlost. Poté se dá graf vykreslit detailněji pomocí „Fine Paint“ tlačítka.

Matematická část programu (polopatě, tu psal táta):

* **compileBoolExpression()** – metoda používá metodu **parseNumExpression()** – která rozparsuje výraz na jednotlivé funkce a poté je z těchto funkcí vyroben algoritmus, podle kterého se relace počítá. Pokuď má výraz špatný syntax, tak metoda vyhodí exception. **compileBoolExpression()** vrací v exception zprávu s přesným popisem chyby v syntaxu, na jaké pozici ve výrazu je chyba a jaká.
* **evalBoolExpression(x,y,pixelsize) –** metoda obsahuje metodu **CalculateReal(x,y,pixelsize) –** rozparsovaný výraz na funkce (každá část výrazu je jednotlivá metoda – ty jsou deklarovány v knihovně **eval\_operation**), tyto jednotlivé operační funkce tvoří poté algoritmus, pomocí kterého **CalculateReal(x,y,pixelsize)** vypočítá jestli pro dané hodnoty x a y výraz platí, při tomto počítaní je užíván zásobník (na ukládání hodnot po každé operaci), který je deklarovaný v knihovně **eval\_stack**. V metodě je ještě jeden argument – *pixelsize*, který je použit na druhý typ počítání funkcí – „Better Functions“. Pokuď je pixelsize -1, tak je používán defaultní algoritmus na počítání.