Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Projekt języka graficznego 2D

Oskar Kocjan
Stanisław Światłoch
Szymon Auguścik

1 Opis języka

Celem naszego projektu jest stworzenie prostego języka graficznego 2D. Podstawową funkcjonalnością będzie tworzenie kształtów, takich jak proste, koła, kwadraty, wielokąty, oraz wykonywania na nich operacji zmiany barwy, przesuwania, skalowania i rotacji. Dodatkowo, zostaną zaimplementowane tablice, pętle, instrukcje warunkowe oraz podstawowa arytmetyka.

2 Składnia

2.1 Tworzenie obiektów

- punkt - point Name: X, Y
- koło - circle Name: CentralPoint, Radius
- odcinek - segment Name: BeginPoint, EndPoint
- wielokąt - polygon Name: GroupOfPoints

2.2 Podstawowe polecenia

2.2.1 Inicjalizacja

canvas: X, Y, Color
Color jest stałą, np. #red, #green, #transparent itd.

2.2.2 Rysowanie

draw Shape
Shape to dowolny kształt

2.2.3 Przypisanie/kopiowanie

assign Name: ...
... to nazwa zmiennej o tym samym typie co Name lub wyrażeniem arytmetycznym, jeżeli Name jest typu num.

2.2.4 Wypisywanie na konsole

- log Name
  Name to nazwa zmiennej
- log [wyrażenie arytmetyczne]
- log "napis"

2.2.5 Zapisywanie obrazu

- save
  Zapisuje aktualny stan płótna do pliku png o tej samej nazwie co skrypt
- save "nazwa_pliku"
  Zapisuje do pliku o podanej nazwie. W przypadku braku rozszerzenia, domyślnie tworzony jest png. Dozwolone rozszerzenia to .png, .jpeg, .bmp i .tga. W nazwie pliku nie można używać pustych spacji, ani żadnego z tych znaków:

```
#%&{}\<>*?/$!'":@+`|=
```

2.3 Grupowanie

group<*type*> Name: MemberNames

- *type* - słowo klucze, do wyboru z puli:
  - drawables
  - circles
  - polygons
  - segments
  - points

  W przypadku użycia *type* jako drawables, możliwe jest utworzenie grupy składającej się z różnych dostępnych typów (Point, Segment, Circle, Polygon). W innym wypadku dozwolone jest tylko użycie zmiennych o typie zgodnym z *type*
- Name - nazwa grupy
- MemberNames - rozdzielone przecinkami nazwy zmiennych wchodzących w skład grupy (cn. 1 nazwa jest konieczna)

2.4 Typy

Dostępne typy

1. point, circle, polygon, segment - rodzina podstawowych kształtów, które mogą zostać narysowane przez użytkownika
2. num - liczba zmiennoprzecinkowa
3. group - struktura przechowująca kilka wartości/obiektów

Pomimo, że num jest typu zmiennoprzecinkowego, w przypadku kiedy potrzebna jest wartość całkowita, domyślnie brana jest tylko część całkowita liczby. Istnieją także odpowiednie operatory zaokrąglenia (opisane poniżej).

2.5 Operatory

- +, -, *, / – podstawowe operatory arytmetyczne
- <=, >, >=, <, =, != – relacje logiczne
- ! – logiczna negacja
- % – modulo
- ˆ, _, ∼ – operatory zaokrąglenia, odpowiednio: w górę, w dół, do najbliższej wartości
- &, | – operatory koniunkcji i alternatywy logicznej

Ponieważ w języku nie ma typów logicznych, za prawdę logiczną przyjmuje się wartość różną od zera, a za fałsz wartość 0. Operatory logiczne zwracają wartość 0, albo 1.

2.6 Rodzaje transformacji

- fill Shape: Color - wypełnienie zadanym kolorem
- move Shape: X, Y - przesunięcie o wektor [X; Y]
- place Shape: P - przesunięcie figury z pierwszego charakterystycznego punktu do punktu P
- rotate Shape: Angle, P - rotacja względem punktu P
- scale Shape: Rate - skalowanie

2.7 Instrukcja iteracyjna

loop Iter start Start until End step Step then

    ...
end

- Iter - nazwa iterowanej zmiennej
- Start, End - wartości: początkowa i końcowa
- Step - krok iteracyjny
- ... - dowolna liczba operacji

2.8 Instrukcja warunkowa

check Cond1 then
    ...
else check Cond2 then
    ...
else then
    ...
end

- Cond1, Cond2 - wyrażenie logiczne lub arytmetyczne (wartość 0 dla fałszu, różne od 0 dla prawdy)
- ... - dowolna liczba operacji

2.9 Komentarze

Język zawiera dwa rodzaje komentarzy:

- wielolinijkowe - "-* *-"
  - komentarz*
  komentarz
  *komentarz *-*
- jednolinijkowe - "--"
  -- komentarz

3 Przykład użycia

3.1 Inicjalizacja zmiennych

- point P1: 1, 3
- num Radius: 2
- circle Circle: P1, Radius
- point P2: 3.1, 6.6
- point P3: 5.6, 10.2
- group Points: P1, P2, P3
- polygon Triangle: Points
- segment Seg: P1, P2
- group Shapes: Circle, Triangle
- iterator I: 5

3.2 Wywołanie transformacji

- fill Square: #red
- draw Shapes[I]

3.3 Zastosowanie instrukcji iteracyjnej

```
loop I start 0 until 4 step 3
    draw Shapes[I]
end
```

3.4 Zastosowanie instrukcji warunkowej

```
check I%2=0
    scale Shapes[I]: 2
    draw Shapes[I]
else check I%3=0
    point Point: 2, 1
    rotate Shapes[I]: 30, Point
    draw Shapes[I]
else
    fill Shapes[I]: #yellow
    draw Shapes[I]
end
```

3.5 Przykładowy program tworzący szachownicę

```
num X: 100
canvas: X, X, #white

num SqrLen: X/8
point P0: 0, 0
point P1: SqrLen, 0
point P2: 0, SqrLen
point P3: SqrLen, SqrLen
group Points: P0, P1, P2, P3
polygon Sqr: Points
fill Sqr: #black

loop I start 0 until 7 step 1
    loop J start 0 until 7 step 1
        check (I+J)%2 = 1
            point P: I*SqrLen, J*SqrLen
            place Sqr: P
            draw Sqr
        end
    end
end
```