Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.

David Šmíd

Ve své práci používám 3D tiskárnu, která je hlavním nástrojem v pseudovědecké laboratoři, kde se zpracovávají hodnoty získávané ze vzduchu. Tiskárna zde funguje jako transformátor zachycených dat, jejichž vizualizované formy tiskne – převádí do 3D reálného objektu.

Pro fungování přístroje stačí prvotní impulz, spuštění programu tiskárny, který pomocí radiopřijímačů zachytává data z elektrosmogu. Získaná data vytvoří virtuální objekt,který je následně poslán do tiskárny k realizaci.

Primární funkce tiskárny je plně využita, ovšem nezanedbatelný je i doprovodný zvukový projev pracujících elektromotorů.

Návštěvník laboratoře může zkoumat vizualizovaný zvuk tiskárny nebo poslouchat zvukový projev vznikající při realizaci vizualizace. Jsou to propojené procesy, na jejichž začátku stojí radiopřijímač, kterému jako zdroj informací slouží elektrosmog. Elektrosmog je všude přítomný, reálný a přitom velmi abstraktní, málokdo si uvědomuje, kolik informací v sobě nese. V laboratoři se z amorfního abstraktního prostředí stává reálný objekt.

Ačkoliv transformace je hlavním zájmem laboratoře, vizualizace, reálný objekt a zvukový projev jsou nedílnou součástí celého komplexu.

Nezanedbatelným bodem je pro mě i fakt, že získávaná data přímo neovlivňuji, ale přijímače tiskárny je zachytí samy, podle kritérií. Lidský faktor ve výsledku není patrný.

Koncept díla spočívá v nikdy nekončící vědecké práci, tudíž se nedá vystavit ve finální podobě, ale jen jako okamžik v danou dobu, kdy výzkum probíhá. Jedním z konečných produktů jsou vytištěné 3D objekty, což jsou transformovaná a realizovaná data ze vzduchu.

TECHNIKA jako prostředek UMĚNÍ

Touha po poznání, puzení – to je společný jmenovatel spojující tak rozdílně nahlížené společenské obory – vědu a umění. Přestože se umělci těší zájímavé společenské pozici a vědci zůstávají nepoznaní, jsou jejich představitelé považováni za nositele a tvůrce kultury.

Z historického hlediska jsou oba obory, dalo by se říci, příbuzné, mají stejné kořeny. Řecké slovo *techné* má stejný význam jako latinské *ars*, přičemž obě slova jsou definována významy: zručnost, znalost, dovednosti získané přemýšlením a bádáním.

Proces přemýšlení, bádání a poznávání tvoří tedy základ oběma táborům. Ještě před hledáním odpovědi si však musíme položit otázku. Vědec se odpovědí na otázku snaží odhalit a dokázat nové zákony a principy, příčinné souvislosti, jež je následně možné považovat za platnou teorii. Naproti tomu umělec pracuje často s abstraktními pojmy, souvislostmi, obrazy skutečností, s metaforami, které následně transformuje do osobních a kulturně historických odkazů a idejí.

Historické zázemí mají tedy obě disciplíny společné. Ať více či méně, vzájemně se ovlivňují a inspirují dodnes, nehledě na jejich dnešní koncepci.

Není to jen doménou současných, moderních umělců se ve vědě inspirovat a přenášet tak vizuální či konceptuální podněty do svých prací.

Rembrandtovo dílo – *Anatomie doktora Tulpa* je fascinováno lékařskou vědou. Z pohledu tehdejšího člověka, zkoumajícího umělce/vědce, se jedná o neprobádanou oblast. Vhodnou k dalším objevům, experimentům a odhalování. Role umělce i vědce, zdá se, jde v tomto směru ruku v ruce. Např. správnost poznatků tehdejší anatomie záležela do značné míry na jejich popisu – tedy na schopnosti reálně zobrazit a zachytit skutečnou strukturu. Což bylo doménou tehdejších „bádajících“ umělců, kteří se účastnili různých veřejných i neveřejných lékařských/bádavých zákroků … nebo je sami prováděli. Poznatky tehdejší doby jsou též limitováni. Co by Rembrandt zobrazoval v dnešních dnech pokroku? …Marta de Menezes, současná umělkyně, využívá dosažené vědění v oblasti biotechnologie. Pod jejíma rukama tak vznikají jedinečná díla – zásahem do vývojových mechanizmů „vytváří“ motýly s křídly zbarvenými tak, jak se v přírodě nevyskytují.
Leonardo Da Vinci dokonale snoubí oba obory s cílem poznat a poznáním obohacovat lidský život.
Objevení atomů přináší zájem o vnitřní mikrosvěty a představy o nich. Newtonova *Optika* okouzluje malíře při zkoumáni tvarů a barev předmětů. V Goetheově *Pojednání o barvách* jsou barvy rozděleny na teplé a studené, popisovány účinky na psychiku… Dílo francouzského matematika Henryho Julese Poincareeho prý mělo vliv jak na Einsteina a jeho pozdější teorii relativity, tak na Pabla Picassa. Picasso ovlivněn existencí časoprostoru hledal novou formu jeho výtvarného vyjádření – *Avignonské slečny*, v plochém prostoru obrazu se vyskytuje čas (časoprostor) v podobě rozdílně znázorněných obličejů postav. → Kubizmus (Cézanne, Seurat – geometrická abstrakce. Něco mezi hudbou a algebrou; Vasilij Kandinskij – kompozice s barvami a tvary. Komponuje je jako hudbu; Mondrian, Kupka – zkoumají vlastnosti světla, skládání barev, vztahy barvy a světla/barvy a rytmu.)

V současné době vznikají projekty spojující vědu a umění v jeden celek; společenství, místa, kde se setkávají vědci a umělci za cílem sdílení informací, vzájemné pomoci a tvorby tzv. *mezioborové laboratoře* (např. projekty účastnící se pražského bienále ENTER). Odtud slovní spojení „umělec/umělci v laboratořích“. Např. Snové reakce BZ – umělecká díla vytvořená v laboratoři; kdy chemické reakce postupem času vytvářejí vizuální fraktální produkt,…

Věda a umění se v různých uměleckých směrech od sebe oddalují, neoddalují se však nikdy natolik, že by se dalo říci, že v určitém časovém období spolu nemají nic společného.

Jsou tedy věda a umění oddělené světy? Jsou oba směry vůči sobě v opozici? Jako emoce a racionalita? Umění a vědu spojuje tvůrčí proces, experimentování, touha po odhalení nového, posunout hranice. Uměleckým postupem tak lze nacházet jiné způsoby řešení vědeckých otázek a obrá-

ceně. Rozdíl mezi oběma světy pak shledáváme v tom, jak je vnímáme rozdílně, jakou pro nás mají hodnotu (s rozdílnou lehkostí je pak hodnotíme, odsuzujeme nebo přijímáme).

Ve chvíli, kdy se vědecká činnost stává součástí uměleckého procesu, tj., kdy umělec přejímá metody výzkumníka, je umělecké dílo vrženo do nového světla souvislostí. Pak se prostory vyhrazené pro výzkum stávají novými galeriemi a výstavními prostory.

Metody a klíčová slova charakterizující projekt:

- druhořadé a skryté vlastnost či doprovodné jevy zaujímají hlavní bod zájmu – akustické projevy tiskárny závislé na rádiových vlnách v její blízkosti jsou základní kámen pro tvorbu virtuální i reálné sochy.
- instrumentalizace stroje (konstrukce tónů, objektů; definice prostoru) – přenos osvojených dovedností a přístupů do oblasti neočekávané, nepříliš běžné. (v tomto případě) Přenesením zvukově aktivního objektu do sféry hudby …získáme nový způsob nahlížení na doposud známou oblast a postupy. Můžeme získat jak v první řadě nové zvuky, tak jevy, které jsou s fungováním a používáním nového stroje spjaty. Ať už se jedná o „notový“ zápis, způsob komponování …či sensuální vjemy. Náhled na stroj je nutně determinován oblastí, ze které na něj nahlížíme, přesto nám však nastiňuje nové možnosti. Některé objevíme až poté, co jej začneme zkoumat a používat, jiné jsou na první pohled zřejmé.

Různé druhy nástrojů mají svoji specifickou pozici danou svými hudebními vlastnostmi a kvalitami. Kvalitami, které připisujeme zvuku, jež vydávají, nikoliv konkrétní konstrukci nástroje (buben – rytmický nástroj, většinou slouží k udávání a udržování tempa, housle – jsou schopné hrát dlouhé a táhlé tóny, mají svoji specifickou barvu, …). Instrumentalizace jakéhokoli předmětu nám tak poskytuje novou paletu pojmů, které si můžeme osvojit, poznat a používat.

Hudební nástroj je spjat s hudbou, se zvuky. U objektu, který je vržen do světa hudebních kritérií ovšem někdy nelze mluvit jako o hudbě, ale spíše hluku (i takovéto nástroje a jejich koncept mají své místo v historii hudby a umění).

Zvuk je ve své podstatě vlnění schopné šířit se různým hmotným prostředím; a to, zda jej nazveme hudbou či hlukem záleží jak na kultuře, ve které žijeme, tak na vlastním subjektivním hodnocení. Náš nejobvyklejší způsob vnímání zvuku je zprostředkován přenosem ve vzduchu. Zdroj zvuku rozechvívá prostředí → zvukové vlny se jím šíří → až k našemu sluchovému aparátu. Vzduch však není jediné médium, které je schopné zvuk přenášet; v instalaci Water Whistle Maxe Neuhause je médiem voda – posluchači proto musí ponořit hlavu pod vodu, aby slyšeli tóny, které jsou skrz vodu reprodukovány (zvuk je jeden ze základních elementů lidského života). Sluch je základní smysl; slouží kromě zraku a hmatu k orientaci v prostoru. Sluchový orgán (Cortiho orgán) je schopen zvuk vnímat dvěma způsoby. Jednak je to tzv. převodní vedení (ušní boltec → sluchovod…) a druhou možností je kostěné vedení, kdy zvuk rozechvívá kosti lebky → až do vnitřního ucha.
Zvukový prožitek tak nemusí být nutně spjat se sluchovým ústrojím (hudbou šířenou vzduchem), ale může se přesunout do oblasti větší tělesnosti a vnímání vibrací. „Poslechnout“ pak může znamenat „prožít“.
- aleatorická, experimentální hudba, doplněná o další rozměr (vizuální)
- technická atraktivnost – souvisí s použitím současných postupů, materiálů, odkazuje se na současná živá témata. Technická atraktivnost často souvisí s datem zavedení technického řešení (zda-li je řešení zastaralé). Oblast technických pokroků se těší velkému zájmu. Ať už to jsou nové produkty - věhlasných značek, z oblasti ekologických přístupů, automobilek, či pouhá konstrukční řešení použitých součástí.
- zkoumat neznámé – klást si otázku …a na ni odpovědět, prohlédnout záměr a odhalit skrytý systém. Člověka motivuje snaha umět zařadit, umět se vypořádat s něčím novým. Nerozumíme-li něče-

mu, konstruujeme teorie, které se posléze snažíme potvrdit – zvídavost. Každý živý tvor, pokud překoná strach, má snahu zkoumat nové. Pro nás může „nové“ znamenat třeba jen jiný kontext…
- působení na smysly – čitelná rovina zvuku (v tomto případě) zapojuje jen jeden z pěti smyslů člověka. Vizuální stránka doplňuje a dokresluje celkovou strukturu a umocňuje celistvý vjem. (Ať se jedná o celkovou technickou kompozici, či vznikající virtuální sochu na monitoru.)

Projekt *Stanice V2S* je jakýsi vizuálně hudební objekt. (Název odkazuje do oblastí nějakého zkoumání, zapisování a strukturování dat). Nejedná se o demonstraci techniky, ač je jejím zájmem, aby vše fungovalo právě tak, jak má. V popředí projektu je slovo i nástroj – tiskárna. Je to zařízení, které zaznamenává nějaká uživatelsky definovaná data (její primární funkce), hlavní zájem však není kladen na tuto primární vlastnost – tisknutí, ale na skutečnost upozaděnou – zvuk. Zařízení má unikátní vlastnost – hraje (při vykonávání jednotlivých úkonů vydávají pohybové součásti přístroje zvuk/tóny – chová se jako elektroakustický měnič). Je tedy možné tuto skutečnost primarizovat a následně celý objekt převést do sféry experimentální hudby. Přičemž je proces doplněn o vizuální stránku (a současně vzniklý artefakt).

Tiskárna jakoby naslouchá (přes elektronické a programové síto v podobě počítače) z okolí data formou antén. Ta následně přehrává, „tiskne“ ve formě tónů, které je schopna vydávat. Současně se „hraná“ data zaznamenávají a následně vizualizují → vzniká tak jakási virtuální socha za doprovodu „hudby“, popř. hmotná 3D studie, skutečná socha, konstruovaných dat - celistvý vjem. Pro diváka je to zhmotňování reality, kterou nevidí, možná ani nevnímá; přesto tu je (v podobě elektromagnetického smogu již od 19. století).

Je velice zajímavé tvořit „z ničeho něco“, velice zjednodušeně, jsou to informace – elektrosmog, který kolem nás je a my ho vůbec nevnímáme. Je zajímavé vyzdvihnout všední věc, neviditelnou skutečnost, přesto každodenní a všeprostorovou, a dát ji vyniknout.

Celý projekt je jakýsi výřez místa, času a limitů – není možné objektivně zobrazit vše. Je tedy nutné pracovat jako vědec; pro zjednodušení vytvořit model, na který se dají teorie aplikovat a který je schopný danou část skutečnosti objektivně formulovat a popsat. Je to jakýsi jazyk, struktura aplikovaná na spojitou skutečnost (realitu), schopný sdělit určitou část skutečnosti, objektivizovat ji a zaznamenat. Proč je to tak? Jako při popisu jakékoliv skutečnosti je nutné zavést názvosloví, pravidla a hranice… a nakonec potvrdit nebo vyvrátit, na základě námi dané struktury, model (abstrahovaný, zjednodušený popis skutečnosti).

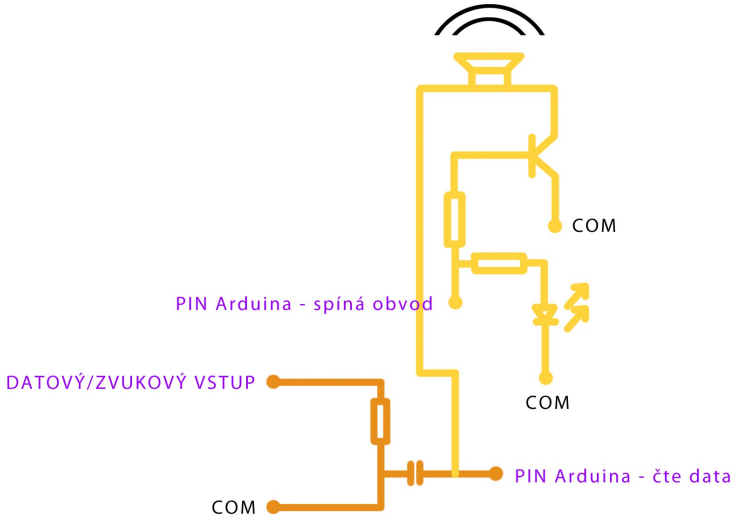
Od seriózního vědeckého projektu se liší tím, že se nesnaží něco nového dokázat, potvrdit nebo objevit a dále s dosaženým výsledkem nějak pracovat. Pouze užívá odvozené názvosloví a techniky; je to jakási hra… využívající svého názvu k umocnění své serióznosti a vážnosti.

Instrument zde funguje jako prostředek k rozšíření našeho vnímání a poznání. Pracuje s prostorem a časem, na kterém je závislý. Výsledný „obraz“ tak je stále jiný, neopakovatelný a prchavý.

OBJEKT JE SLOŽEN Z ČÁSTÍ:

-RÁDIOVÁ ČÁST – Tu tvoří 3 rádiové přijímače přijímající data z FM/AM frekvenčních pásem a programovatelná deska Arduino, která se stará o přenos dat po sériové lince do PC. Celá rádiová část je napájena z USB portu počítače (odběr cca. 50mA/radiopřijímač; napětí 5V je pro bezproblémový chod dostačující).

SCHÉMA:



Výstup ze zesilovače je zprostředkován jednoduchým obvodem; 8Ohm rezistor, 10uF kondenzátor. Činnost programu je signalizována LED a současně jsou sbíraná data reprodukována (cca 1 vteřina).

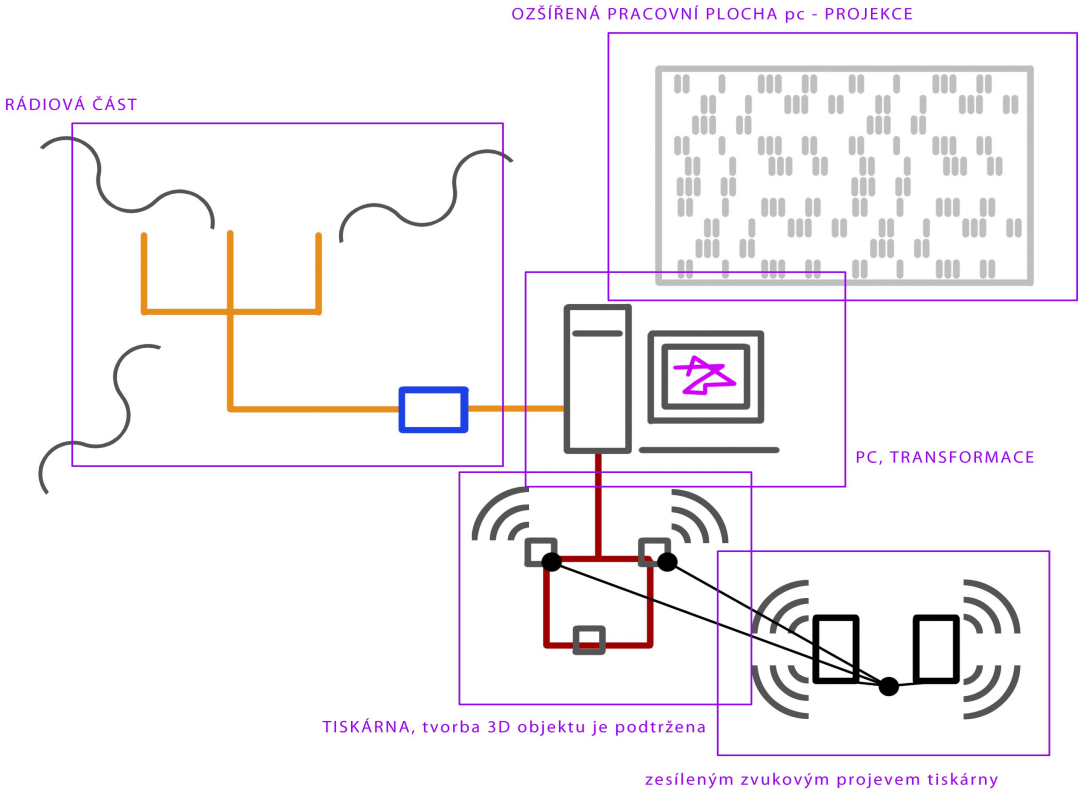
SOFTWARE ARDUINA:
-str. 9-

-PC – Zpracovávající data, vizualizuje a zároveň i vytváří zálohu zpracovávaných dat.

SOFTWARE:
-str. 10-
Program v Python GUI - stará se o komunikaci s Arduinem ovládajícím rádia; dále získaná data ukládá a sděluje vizualizačnímu programu, že data jsou již dostupná.

SOFTWARE:
-str. 13-
Program v Blender prostředí - generační a vizualizační program; program sestavuje nový objekt na základě řady dat, jakési DNA objektu, následně je v prostředí Blenderu tento objekt vizualizován.

-3D TISKÁRNA – „Hraje“, popř. tiskne.
(Firmware 3D tiskárny je uživatelsky nastavený pro její bezproblémový chod.)
Pro reprodukci a zesílení zvuků tiskárny jsou použity následující prvky: piezo snímače → reprozesilovač jako předzesilovač → koncová reprosoustava (s dalším zesílením)


OZŠÍŘENÁ PRACOVNÍ PLOCHA pc - PROJEKCE
RÁDIOVÁ ČÁST
PC, TRANSFORMACE
TISKÁRNA, tvorba 3D objektu je podtržena
zesíleným zvukovým projevem tiskárny

Program po vyžádání – objeví-li se na vstupu -1, vypočítá frekvence na jednotlivých vstupních pinech a po sériové lince odešle.

Nastavení proměnných.

```
int pin_1 = 5;
int pin_2 = 6;
int pin_3 = 7;
int pin_4 = 8;
String incomingData = "";
```

inicializace sériové linky a pinů Arduina.

```
void setup() {
	Serial.begin(115200);
	pinMode(pin_1, INPUT);
	pinMode(pin_2, INPUT);
	pinMode(pin_3, INPUT);
	pinMode(pin_4, OUTPUT);
}
```

Hlavní smyčka programu. Kontroluje příchozí data a odesílá sérii 52 čísel – frekvence.

```
void loop() {
	if (Serial.available() > 0) {
		incomingData = Serial.read();
		if (incomingData == 49){
			digitalWrite(pin_4, HIGH);
			#define COUNT 52
			for(unsigned int j=0; j<COUNT; j++) {
				int sens_value_1 = getFrequency(pin_1);
				Serial.print(sens_value_1);
				delay(0.1);
				Serial.print(" ");
				delay(0.1);
				Serial.flush();
			}
			digitalWrite(pin_4, LOW);
		}
	}
}
```

Funkce počítající frekvenci na definovaném pinu (8 vzorků – pro dostatečně rychlou odezvu).

```
long getFrequency(int pin) {
	#define SAMPLES 8
	long freq = 0;
	for(unsigned int j=0; j<SAMPLES; j++) freq+= 500000/pulseIn(pin, HIGH,
25000);
}
```

Import standardních modulů z Pythonu (2.6).

```
import serial
import time
```

Při inicializaci třídy se načítá počáteční hodnota ze sériového portu, aby nedocházelo k chybě, kdy je funkce volaná poprvé a vrácená hodnota je prázdný seznam.

```
class ReadArduinoSerial:
    def __init__(self, port, baud):
        self.ser = serial.Serial(port , baud, bytesize=8, parity='N',
stopbits=1, timeout= .5)
        time.sleep(5)
        r = self.read()
        print ("ASR: zkusebni cteni ze serioveho portu: " + str(r))
        print ("ASR: connected to Arduino")
```

Data na sériový port jsou posílána na „požádání“ - objeví-li se na vstupu – 1.

```
    def read(self):
        self.ser.write(1)
        data = self.ser.readline()
        self.ser.flush()
        return data
        print data
```

Funkce pro uvolnění sériového portu.

```
    def disconnect(self):
        self.ser.close()
        print ("ASR: ArduinoRead port closed")
```

//python program komunikující s Arduinem; ukládá data do textového souboru

Import modulů z Pythonu.

```
import math
import time
import ArduinoSerialReceiver_pyth26
import sys
import random
```

Inicializace komunikace.

```
class Freq_to_Sculpture:
    def __init__(self):
        print ("main: Freq_to_G initializing..." + "\n")
        self.receiver = ArduinoSerialReceiver_pyth26.ReadArduinoSerial("com23",
115200)
```

Funkce pro čtení dat.

```
    def read(self):
        data = self.receiver.read()
        return data
```

V této funkci se kontroluje předdefinovaný počet načtených dat.

```
    def data(self):
        data = self.read().split()
        while len(data) < 52:
            data = self.read().split()

        for i in range (48, 52):
            print("data[i]", data[i])
            if float(data[i]) > 1000:
                data[i] = 1
            else:
                data[i] = 0

        print("data", data, len(data))
        self.write_to_file(data)
```

Zápis dat pro závislý program, funkce přepíše některé textové soubory – dá tak čekajícímu programu „vědět“, že data jsou již připravena a zároveň nastaví čekající program na počáteční hodnoty .

```
    def write_to_file(self, data):
        try:
            goal = open("D:/praace/pokusy/rainbow/new_data.txt", "w")
            store = open("D:/praace/pokusy/rainbow/store.txt", "a")
        except:
            print ("new_data/store.txt ...bussy" + "\n")

        for item in data:
            goal.write(str(item) + " ")
            store.write(str(item) + " ")
        for i in range(2):
            store.write("\n")
        goal.close()
        store.close()
        print("lenght of list:", len(data))
```

Základní smyčka programu.

```
    def run(self):
        run = True
        try:
            while run == True:
                time.sleep(10)
                try:
                    variable = open("D:/praace/pokusy/rainbow/var.txt", "r")
                except:
                    print("no variable")

                x = variable.read()
                variable.close()

                if int(x) == 1:
                    print("...processing")
                    try:
                        variable = open("D:/praace/pokusy/rainbow/var.txt", "w")
                    except:
                        print ("variable.txt ...bussy" + "\n")

                    variable.write("0")
                    variable.close()

                    self.data()

                    try:
                        start = open("D:/praace/pokusy/rainbow/start.txt", "w")
                        first_read =
open("D:/praace/pokusy/rainbow/first_read.txt", "w")
                        counter = open("D:/praace/pokusy/rainbow/counter.txt",
"w")
                    except:
                        print ("start/first_read/counter.txt ...bussy" + "\n")

                    start.write("1")
                    first_read.write("1")
                    counter.write("0")
                    start.close()
                    first_read.close()
                    counter.close()
                    print("beginning set")
                else:
                    print("...waiting for start command")

        except KeyboardInterrupt:
            print ("main: shutting the program down")
            run = False
            print ("main: program stopped")

fts = Freq_to_Sculpture()
fts.run()
```

Import důležitých modulů.

```
from bpy import *
import bpy
import math, mathutils, random
```

Nastavení základních proměnných

```
scene = bge.logic.getCurrentScene()
cont = bge.logic.getCurrentController()
own = cont.owner
```

Vytvoření základních barevnostních shchémat – palet.

```
mat_bila = bpy.data.materials['bila_n']
mat_seda = bpy.data.materials['seda_n']
mat_cerna = bpy.data.materials['cerna_n']
mat_bila_alpha = bpy.data.materials['bila_alpha_n']
paleta_1 = [mat_bila,  mat_seda, mat_cerna, mat_bila_alpha]

mat_zelena = bpy.data.materials['zelena_n']
mat_hneda = bpy.data.materials['hneda_n']
paleta_2 = [mat_zelena, mat_hneda]

mat_morph = bpy.data.materials['morph_1']
paleta_3 = [mat_morph]

mat_fialova = bpy.data.materials['fialova_n']
paleta_4 = [mat_fialova]

mat_biilaa = bpy.data.materials['biilaa']
mat_modra = bpy.data.materials['modra_n']
paleta_5 = [mat_biilaa, mat_modra]

mat_oranzova = bpy.data.materials['oranzova_n']
mat_bila2 = bpy.data.materials['bila_n2']
paleta_6 = [mat_oranzova, mat_bila2, mat_modra, mat_biilaa]

paleta_7 = [mat_bila2]

palety = [paleta_1, paleta_2, paleta_6, paleta_7] # paleta_3, paleta_4,
paleta_5,
```

Načtení dat pro tvorbu objektu.

```
####################nacteni dat pro morphing#####################################
try:
    goal = open("D:/praace/pokusy/rainbow/new_data.txt", "r")
    print ("data.txt opened")
except:
    print ("data.txt ...bussy" + "\n")
data = goal.read().split()
goal.close()
#print("data", data)
```

Protože se program v Blenderu několikrát restartuje – pro překreslení scény, je nutné, aby program začínal z určitého místa. O to se starají poznámky programu v textových souborech, které program při spouštění čte.

```
####################first_read#################################################
try:
    first_read = open("D:/praace/pokusy/rainbow/first_read.txt", "r")
```

```
except:
    print ("first_read.txt ...bussy" + "\n")
own["first_read"] = int(first_read.read().split()[0])
first_read.close()

try:
    first_read = open("D:/praace/pokusy/rainbow/first_read.txt", "w")
except:
    print ("first_read.txt ...bussy" + "\n")

first_read.write("0")
first_read.close()
```

Záznam/načtení probíhající fáze

```
##################counter#####################################################
try:
    count = open("D:/praace/pokusy/rainbow/counter.txt", "r")
except:
    print ("counter.txt ...bussy" + "\n")
own["counter"] = int(count.read().split()[0])
count.close()
```

Informace o aktuální barevné paletě

```
##################paleta######################################################
if own["first_read"] == 0:
    try:
        pal = open("D:/praace/pokusy/rainbow/paleta.txt", "r")
    except:
        print ("paleta.txt ...bussy" + "\n")
    rp = own["paleta"] = int(pal.read())
    paleta = palety[rp]
    pal.close()
else:
    rp = random.randint(0, len(palety)-1)
    paleta = palety[rp]
    p_l = len(paleta)

    try:
        pal= open("D:/praace/pokusy/rainbow/paleta.txt", "w")
    except:
        print ("paleta.txt ...bussy" + "\n")

    pal.write(str(rp))
    pal.close()
##############################################################################
```

Funkce v níž se načtená data „roztřídí“.

```
def execute():
    dev_BA = 100
    dev = 200.00

    Body_Len = round(float(data[0])/dev_BA)
    Body_Phase = round(float(data[1])/dev_BA)
    Body_Step = round(float(data[2]))/dev
    Body_Amplitude = (float(data[3]))/5.0/dev
    Body_StepAngle = float(data[4])/dev
    Body_ModAmplitude = (float(data[5]))/20.0/dev
    Body_ModStep = (float(data[6]))/5.0/dev
    Body_Joint = 0

    A_minA_ = (float(data[35]))/50.0
```

```
A_maxA_ = (float(data[36]))/25.0
if A_minA_ > A_maxA_:
    A_minA = A_maxA_
    A_maxA = A_minA_
else:
    A_minA = A_minA_
    A_maxA = A_maxA_
A_minL_ = (float(data[37]))/200.0
A_maxL_ = (float(data[38]))/100.0
if A_minL_ > A_maxL_:
    A_minL = A_maxL_
    A_maxL = A_minL_
else:
    A_minL = A_minL_
    A_maxL = A_maxL_

H_minA_ = (float(data[39]))/50.0
H_maxA_ = (float(data[40]))/25.0
if H_minA_ > H_maxA_:
    H_minA = H_maxA_
    H_maxA = H_minA_
else:
    H_minA = H_minA_
    H_maxA = H_maxA_
H_minL_ = (float(data[41]))/200.0
H_maxL_ = (float(data[42]))/100.0
if H_minL_ > H_maxL_:
    H_minL = H_maxL_
    H_maxL = H_minL_
else:
    H_minL = H_minL_
    H_maxL = H_maxL_

L_minA_ = (float(data[43]))/50.0
L_maxA_ = (float(data[44]))/25.0
if L_minA_ > L_maxA_:
    L_minA = L_maxA_
    L_maxA = L_minA_
else:
    L_minA = L_minA_
    L_maxA = L_maxA_
L_minL_ = (float(data[45]))/200.0
L_maxL_ = (float(data[46]))/100.0
if L_minL_ > L_maxL_:
    L_minL = L_maxL_
    L_maxL = L_minL_
else:
    L_minL = L_minL_
    L_maxL = L_maxL_

Balls = int(data[35])
Ball_Size = float(data[36])/30/dev
Ball_Hairy = int(data[37])
Rand_Hairy = int(data[38])
Ball_Top = int(data[39])

Arm_Len = round(float(data[7])/dev_BA)
Arm_Phase = round(float(data[8])/dev_BA)
Arm_Step = round(float(data[9]))/dev
Arm_Amplitude = (float(data[10]))/15.0/dev
Arm_StepAngle = float(data[11])/dev
```

```
    Arm_ModAmplitude = (float(data[12]))/20.0/dev
    Arm_ModStep = (float(data[13]))/5.0/dev
    Arm_Joint = 0

    Leg_Len = round(float(data[14])/dev_BA)
    Leg_Phase = round(float(data[15])/dev_BA)
    Leg_Step = round(float(data[16]))/dev
    Leg_Amplitude = (float(data[17]))/15.0/dev
    Leg_StepAngle = float(data[18])/dev
    Leg_ModAmplitude = (float(data[19]))/20.0/dev
    Leg_ModStep = (float(data[20]))/5.0/dev
    Leg_Joint = 0

    Horn_Len = round(float(data[21])/dev_BA)
    Horn_Phase = round(float(data[22])/dev_BA)
    Horn_Step = round(float(data[23]))/dev
    Horn_Amplitude = (float(data[24]))/15.0/dev
    Horn_StepAngle = float(data[25])/dev
    Horn_ModAmplitude = (float(data[26]))/20.0/dev
    Horn_ModStep = (float(data[27]))/5.0/dev
    Horn_Joint = 0

    Spin_Len = round(float(data[28])/dev_BA)
    Spin_Phase = round(float(data[29])/dev_BA)
    Spin_Step = round(float(data[30]))/dev
    Spin_Amplitude = (float(data[31]))/5.0/dev
    Spin_StepAngle = float(data[32])/dev
    Spin_ModAmplitude = (float(data[33]))/20.0/dev
    Spin_ModStep = (float(data[34]))/5.0/dev
    Spin_Joint = 0
```

Volání funkce pro generaci těla objektu.

```
    generate(Body_Len, Body_Phase, Body_Step, Body_Amplitude, Body_StepAngle,
Body_ModAmplitude, Body_ModStep, Body_Joint, Balls, Ball_Size, Ball_Hairy,
Rand_Hairy, Arm_Len, Arm_Phase, Arm_Step, Arm_Amplitude, Arm_StepAngle,
Arm_ModAmplitude, Arm_ModStep, Arm_Joint, Leg_Len, Leg_Phase, Leg_Step,
Leg_Amplitude, Leg_StepAngle, Leg_ModAmplitude, Leg_ModStep, Leg_Joint,
Horn_Len, Horn_Phase, Horn_Step, Horn_Amplitude, Horn_StepAngle,
Horn_ModAmplitude, Horn_ModStep, Horn_Joint, Spin_Len, Spin_Phase, Spin_Step,
Spin_Amplitude, Spin_StepAngle, Spin_ModAmplitude, Spin_ModStep, Spin_Joint,
A_minA, A_maxA, A_minL, A_maxL, H_minA, H_maxA, H_minL, H_maxL, L_minA, L_maxA,
L_minL, L_maxL)
```

Funkce pro generaci těla objektu.

```
def generate(body_Len, body_Phase, body_Step, body_Amplitude, body_StepAngle,
body_ModAmplitude, body_ModStep, body_Joint, balls, ball_Size, ball_Hairy,
rand_Hairy, arm_Len, arm_Phase, arm_Step, arm_Amplitude, arm_StepAngle,
arm_ModAmplitude, arm_ModStep, arm_Joint, leg_Len, leg_Phase, leg_Step,
leg_Amplitude, leg_StepAngle, leg_ModAmplitude, leg_ModStep, leg_Joint,
horn_Len, horn_Phase, horn_Step, horn_Amplitude, horn_StepAngle,
horn_ModAmplitude, horn_ModStep, horn_Joint, spin_Len, spin_Phase, spin_Step,
spin_Amplitude, spin_StepAngle, spin_ModAmplitude, spin_ModStep, spin_Joint,
a_minA, a_maxA, a_minL, a_maxL, h_minA, h_maxA, h_minL, h_maxL, l_minA, l_maxA,
l_minL, l_maxL):

    # Extrude One Step ---------------------------------------->
```

Funkce pro změnu polohy, velikosti a rotace pro různé skupiny vertexů.

```
    def transStep(dA, sA, rA, axis_x, axis_y, axis_z):
        bpy.ops.transform.shrink_fatten(value=dA, mirror=False,
proportional='DISABLED', proportional_edit_falloff='SMOOTH',
```

```
proportional_size=1, snap=False, snap_target='CLOSEST', snap_point=(0, 0, 0),
snap_align=False, snap_normal=(0, 0, 0), release_confirm=False)
        bpy.ops.transform.resize(value=(sA, sA, sA), constraint_axis=(False,
False, False), constraint_orientation='GLOBAL', mirror=False,
proportional='DISABLED', proportional_edit_falloff='SMOOTH',
proportional_size=0.0762767, snap=False, snap_target='CLOSEST', snap_point=(0,
0, 0), snap_align=False, snap_normal=(0, 0, 0), texture_space=False,
release_confirm=False)

        # Get the mesh
        me = ob.data
        rotDir = 1
        for p in me.polygons:
            if p.select == True:
                p_center = [0.0, 0.0, 0.0]
                for v in p.vertices:
                    p_center[0] += me.vertices[v].co[0]
                p_center[0] /= len(p.vertices)
                if p_center[0] > 0.0:
                    rotDir = 1.0
                else:
                    rotDir = -1.0

        bpy.ops.transform.rotate(value=(rA*rotDir), axis=(axis_x, axis_y,
axis_z), constraint_axis=(False, False, False), constraint_orientation='GLOBAL',
mirror=False, proportional='DISABLED', proportional_edit_falloff='SMOOTH',
proportional_size=0.0762767, snap=False, snap_target='CLOSEST', snap_point=(0,
0, 0), snap_align=False, snap_normal=(0, 0, 0), release_confirm=False)

    # TransformFaces(Multi Step)
---------------------------------------------------->
```

Funkce určující se kterými skupinami vertexů se bude manipulovat.

```
    def transformFaces(vGroup, mainLevel, stepNum, stepDist, stepRot, startDeg,
stepDeg, modLevel, modRate, axis, jointNum, color):
        # Make sure nothing is selected
        bpy.ops.object.mode_set(mode='EDIT')
        bpy.ops.mesh.select_all(action='DESELECT')
        selPart(vGroup)
        bpy.ops.object.mode_set(mode='OBJECT')

        groupList = []

        for i, f in enumerate(ob.data.polygons):
            if ob.data.polygons[i].select == True:
                newGroup = ob.vertex_groups.new('mygroup')
                groupList.append(newGroup)
                for v in f.vertices:
                    newGroup.add([v], 1.0, 'REPLACE')
        for g in groupList:
            bpy.ops.object.mode_set(mode='EDIT')
            bpy.ops.mesh.select_all(action='DESELECT')
            ob.vertex_groups.active_index = g.index
            bpy.ops.object.vertex_group_select()
            bpy.ops.mesh.select_more()
            bpy.context.object.active_material_index = color
            bpy.ops.object.material_slot_assign()
            print(str(vGroup), "rp", rp, "color", color)#palety[rp][color])
            bpy.ops.mesh.select_all(action='DESELECT')
            ob.vertex_groups.active_index = g.index
            bpy.ops.object.vertex_group_select()
```

```
            bpy.ops.object.vertex_group_remove(all=False)

            t = startDeg
            oldValue = 1.0
            j = 0
            for i in range(stepNum):
                bpy.ops.mesh.extrude_region(mirror=False)
                if not i:
                    bpy.ops.object.vertex_group_remove_from(all=True)

                bpy.ops.object.mode_set(mode='OBJECT')
                bpy.ops.object.mode_set(mode='EDIT')

                newValue = (math.sin(math.radians(t)
+math.sin(math.radians(t/2.3)))+2.0)*mainLevel
                newRatio = newValue/oldValue

                dA = stepDist
                sA = newRatio
                rA = modLevel*math.sin(math.radians(t*modRate))*(-stepRot/10)
                axis_x = axis[0]
                axis_y = axis[1]
                axis_z = axis[2]
                transStep(dA, sA, rA, axis_x, axis_y, axis_z)
                if jointNum != 0:
                    print ("jointNum > 0")
                    j = j+1
                    if j == jointNum:
                        bpy.ops.mesh.extrude_region(mirror=False)
                        transStep(-0.01, 0.6, 0, axis_x, axis_y, axis_z)
                        bpy.ops.mesh.extrude_region(mirror=False)
                        transStep(-0.01, 1.66, 0, axis_x, axis_y, axis_z)
                        j = 0
                t = t + stepDeg
                oldValue = newValue

    # transformFaces!(END) ------------------------------------------------<

    # bodyShape ---------------------------------------------------------->
```

Příprava pro tvar těla objektu.

```
    def bodyShape(vGroup):
        newGroup = ob.vertex_groups.new(vGroup)
        bpy.ops.object.mode_set(mode='OBJECT')
        me = ob.data
        for p in me.polygons:
            p_center = [0.0, 0.0, 0.0]
            for v in p.vertices:
                p_center[0] += me.vertices[v].co[0]
                p_center[1] += me.vertices[v].co[1]
                p_center[2] += me.vertices[v].co[2]
            p_center[0] /= len(p.vertices) # division by zero
            p_center[1] /= len(p.vertices) # division by zero
            p_center[2] /= len(p.vertices) # division by zero

            if p_center[1] > -0.1:
                p.select = True
            else:
                p.select = False
```

```
        bpy.ops.object.mode_set(mode='EDIT')
        bpy.ops.object.vertex_group_set_active(group=vGroup)
        bpy.ops.object.vertex_group_assign(new=False)
        print ("bodyShape done")
    # bodyShape(END) ----------------------------------------------------<

    # select faces from Groups ------------------------------------------->
```

Označení/výběr skupiny vertexů.

```
    def selPart(vGroup):
        bpy.ops.object.vertex_group_set_active(group=vGroup)
        bpy.ops.object.vertex_group_select()
        print ("selPart done")
    # select faces from Groups (END)-------------------------------------<

    # cell subdivide ----------------------------------------->
```

Funkce pro rozdělení/rozčlenění povrchu objektu.

```
    def cellSubdiv():
        bpy.ops.object.modifier_add(type='SUBSURF')
        bpy.ops.object.mode_set(mode='OBJECT')
        bpy.ops.object.modifier_apply(apply_as='DATA', modifier="Subsurf")
        print ("cellSubdiv done")
    # cell subdivide (END)-----------------------------------------<

    # assignFacesAngle---------------------------------------------->
```

Funkce pro označení/výběr skupiny vertexů podle kritérií.

```
    def assignFacesAngle(vGroup, fAngleMin, fAngleMax, yMin, yMax):
        newGroup = ob.vertex_groups.new(vGroup)
        bpy.ops.object.mode_set(mode='OBJECT')

        up = mathutils.Vector((0.0,0.0,1.0))

        # Lets loop through all the faces in the mesh
        me = ob.data
        for p in me.polygons:
            p_center = [0.0, 0.0, 0.0]
            for v in p.vertices:
                p_center[0] += me.vertices[v].co[0]
                p_center[1] += me.vertices[v].co[1]
                p_center[2] += me.vertices[v].co[2]
            p_center[0] /= len(p.vertices) # division by zero
            p_center[1] /= len(p.vertices) # division by zero
            p_center[2] /= len(p.vertices) # division by zero
            if p.normal.angle(up) > math.radians(fAngleMin) and
p.normal.angle(up) < math.radians(fAngleMax) and p_center[1] > yMin and
p_center[1] < yMax :
   # Set select to true
                p.select = True
            else:
                p.select = False

        bpy.ops.object.mode_set(mode='EDIT')
        bpy.ops.object.vertex_group_set_active(group=vGroup)
        bpy.ops.object.vertex_group_assign(new=False)
        bpy.ops.mesh.select_all(action='DESELECT')
        print ("assignfacesAngle done")
    # assignFacesAngle(END)----------------------------------<

        # assignFacesAngle---------------------------------------------->
```

Funkce pro náhodné označení/výběr skupiny vertexů.

```
    def assignFacesRandom(vGroup, fAngleMin, fAngleMax, yMin, yMax, extr):
        newGroup = ob.vertex_groups.new(vGroup)
        bpy.ops.object.mode_set(mode='EDIT')
        perc = random.randint(0,5)
        bpy.ops.mesh.select_random(percent=perc, extend=False)
        if extr == True:
            bpy.ops.mesh.extrude_faces_indiv(mirror=False)

#bpy.ops.mesh.extrude_region_move(MESH_OT_extrude_region={"mirror":False},
TRANSFORM_OT_translate={"value":(0,0,0,), "constraint_axis":(False, False,
True), "constraint_orientation":'NORMAL', "mirror":False,
proportional='DISABLED', proportional_edit_falloff='SMOOTH',
proportional_size=1, snap=False, snap_target='CLOSEST', snap_point=(0, 0, 0),
snap_align=False, snap_normal=(0, 0, 0), release_confirm=False})
            bpy.ops.transform.resize(value=(0.3, 0.3, 0.3),
constraint_axis=(False, False, False), constraint_orientation='LOCAL',
mirror=False, proportional='DISABLED', proportional_edit_falloff='SMOOTH',
proportional_size=1, snap=False, snap_target='CLOSEST', snap_point=(0, 0, 0),
snap_align=False, snap_normal=(0, 0, 0), texture_space=False,
release_confirm=False)
        bpy.ops.object.vertex_group_set_active(group=vGroup)
        bpy.ops.object.vertex_group_assign(new=False)
        bpy.ops.mesh.select_all(action='DESELECT')
        print ("assignFacesRamdom done")
```

Funkce pro označení/výběr skupiny vertexů podle kritérií.

```
    def assignFacesBall(vGroup):
        newGroup = ob.vertex_groups.new(vGroup)
        bpy.ops.object.mode_set(mode='EDIT')
        bpy.ops.mesh.select_all(action='DESELECT')
        bpy.ops.mesh.select_random(percent=2, extend=False)
        bpy.ops.object.vertex_group_set_active(group=vGroup)
        bpy.ops.object.vertex_group_assign(new=False)
        bpy.ops.mesh.select_all(action='DESELECT')
        bpy.ops.object.mode_set(mode='OBJECT')
```

Funkce pro přidávání kulovitých objektů.

```
    def addBalls(vGroup, size, hairy, rand_hairy, color):
        me= ob.data
        bpy.ops.object.mode_set(mode='EDIT')
        bpy.ops.mesh.select_all(action='DESELECT')
        selPart(vGroup)
        bpy.ops.object.mode_set(mode='OBJECT')
        groupList = []
        for i,f in enumerate(ob.data.polygons):
            if ob.data.polygons[i].select == True:
                p_center = [0.0, 0.0, 0.0]
                for v in f.vertices:
                    p_center[0] += me.vertices[v].co[0]
                    p_center[1] += me.vertices[v].co[1]
                    p_center[2] += me.vertices[v].co[2]
                p_center[0] /= len(f.vertices)
                p_center[1] /= len(f.vertices)
                p_center[2] /= len(f.vertices)
                groupList.append(p_center)
        bpy.ops.object.mode_set(mode='EDIT')
        for i in range(len(groupList)):
            rand_size = random.uniform(0.02, size)
            bpy.ops.mesh.primitive_ico_sphere_add(subdivisions= 2, size=
rand_size, location= (groupList[i]))
```

```
            c = random.randint(0, len(paleta)-1)
            bpy.context.object.active_material_index = c
            bpy.ops.object.material_slot_assign()

#            if i == 1:
#                hairy = True
#            else:
#                hairy = False

            if rand_hairy == 1:
                h = random.randint(0,10)
                if h == 1:
                    hairy = True
                else:
                    hairy = False

            if hairy == True:
                c = random.randint(0, len(paleta))
                bpy.ops.mesh.subdivide(number_cuts= 1, smoothness= 1, quadtri=
False, quadcorner= "STRAIGHT_CUT", fractal= 0, fractal_along_normal= 0, seed= 0)
                bpy.ops.mesh.extrude_faces_indiv(mirror=False)
                bpy.ops.transform.resize(value=(0.3, 0.3, 0.3),
constraint_axis=(False, False, False), constraint_orientation='LOCAL',
mirror=False, proportional='DISABLED', proportional_edit_falloff='SMOOTH',
proportional_size=1, snap=False, snap_target='CLOSEST', snap_point=(0, 0, 0),
snap_align=False, snap_normal=(0, 0, 0), texture_space=False,
release_confirm=False)
                ob.vertex_groups.new("hairy")
                bpy.ops.object.vertex_group_set_active(group="hairy")
                bpy.ops.object.vertex_group_assign(new=False)
                transformFaces("hairy",spin_Amplitude, spin_Len, 0.2,
spin_StepAngle, spin_Phase, spin_Step, spin_ModAmplitude, spin_ModStep, [-
2.22045e-16, -1, -0], spin_Joint, c)

        bpy.ops.object.mode_set(mode='OBJECT')

    # Generate Object ---------------------------------------->
```

Nastavení prostředí pro práci a orientaci v prostoru.

```
    areas = bpy.context.screen.areas
    for area in areas:
        if area.type == 'VIEW_3D':
            bpy.context.space_data.pivot_point = 'INDIVIDUAL_ORIGINS'
```

Deklarace důležité proměnné.

```
    ob = bpy.context.active_object
```

Program zjišťuje (z textových dokumentů) zda již byl krok vykonán. Následně vykonává po sobě jdoucí příkazy a restatuje zobrazovanou scénu.

```
    if own["first_read"] == 1:
        bpy.ops.mesh.primitive_plane_add(view_align=False, enter_editmode=False,
location=(0, 0, 0), rotation=(0, 0, 0), layers=(True, False, False, False,
False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False))
        bpy.ops.object.mode_set(mode='EDIT')
        bpy.ops.transform.rotate(value=(-1.5708), axis=(-1, -2.22045e-16,
-4.93038e-32), constraint_axis=(False, False, False),
constraint_orientation='LOCAL', mirror=False, proportional='DISABLED',
proportional_edit_falloff='SMOOTH', proportional_size=1, snap=False,
snap_target='CLOSEST', snap_point=(0, 0, 0), snap_align=False, snap_normal=(0,
```

```
0, 0), release_confirm=False)
        bpy.ops.transform.resize(value=(0.05, 1, 0.05), constraint_axis=(False,
False, False), constraint_orientation='GLOBAL', mirror=False,
proportional='DISABLED', proportional_edit_falloff='SMOOTH',
proportional_size=0.0762767, snap=False, snap_target='CLOSEST', snap_point=(0,
0, 0), snap_align=False, snap_normal=(0, 0, 0), texture_space=False,
release_confirm=False)
        bpy.ops.object.mode_set(mode='OBJECT')

        ob = bpy.context.active_object

        for x in range (p_l):
            bpy.ops.object.material_slot_add()
        i = 0
        for item in paleta:
            ob.material_slots[i].material = item
            i += 1
        mats = bpy.context.object.material_slots.__len__()

        bpy.ops.object.mode_set(mode='EDIT')
        bpy.ops.mesh.select_all(action='DESELECT')

        bodyShape('body')

        try:
            count = open("D:/praace/pokusy/rainbow/counter.txt", "w")
        except:
            print ("counter.txt ...bussy" + "\n")
        count.write(str(own["counter"]+1))
        count.close()

        own["restart"] = 1

    if own["counter"] == 1:
        c = random.randint(0, len(paleta)-1)
        transformFaces("body", body_Amplitude, body_Len, 0.5, body_StepAngle,
body_Phase, body_Step, body_ModAmplitude, body_ModStep, [-1, -2.22045e-16, -0],
body_Joint, c)
        bpy.ops.object.mode_set(mode='EDIT')
        bpy.ops.mesh.select_all(action='SELECT')
        bpy.ops.mesh.normals_make_consistent(inside = False)
        bpy.ops.mesh.select_all(action='DESELECT')
        bpy.ops.object.mode_set(mode='OBJECT')
        try:
            count = open("D:/praace/pokusy/rainbow/counter.txt", "w")
        except:
            print ("counter.txt ...bussy" + "\n")
        count.write(str(own["counter"]+1))
        count.close()

        cellSubdiv()

        own["restart"] = 1
        print("body generated")
    else:
        print("body was not generated")

    if own["counter"] == 2:
        assignFacesAngle('arm', a_minA, a_maxA, a_minL, a_maxL)
        assignFacesAngle('horn', h_minA, h_maxA, h_minL, h_maxL)
        assignFacesAngle('leg', l_minA, l_maxA, l_minL, l_maxL)
```

```
        c = random.randint(0, len(paleta)-1)
        transformFaces('arm', arm_Amplitude, arm_Len, -0.2, arm_StepAngle,
arm_Phase, arm_Step, arm_ModAmplitude, arm_ModStep, [-2.22045e-16, -1, -0],
arm_Joint, c)

        try:
            count = open("D:/praace/pokusy/rainbow/counter.txt", "w")
        except:
            print ("counter.txt ...bussy" + "\n")
        count.write(str(own["counter"]+1))
        count.close()

        own["restart"] = 1

    if own["counter"] == 3:
        c = random.randint(0, len(paleta)-1)
        transformFaces('horn', horn_Amplitude, horn_Len, -0.2,horn_StepAngle,
horn_Phase, horn_Step, horn_ModAmplitude, horn_ModStep, [-2.22045e-16, -1, -0],
horn_Joint, c)

        try:
            count = open("D:/praace/pokusy/rainbow/counter.txt", "w")
        except:
            print ("counter.txt ...bussy" + "\n")
        count.write(str(own["counter"]+1))
        count.close()

        own["restart"] = 1

    if own["counter"] == 4:
        c = random.randint(0, len(paleta)-1)
        transformFaces('leg', leg_Amplitude, leg_Len, -0.2, leg_StepAngle,
leg_Phase, leg_Step, leg_ModAmplitude, leg_ModStep, [-2.22045e-16, -1, -0],
leg_Joint, c)

        try:
            count = open("D:/praace/pokusy/rainbow/counter.txt", "w")
        except:
            print ("counter.txt ...bussy" + "\n")
        count.write(str(own["counter"]+1))
        count.close()

        own["restart"] = 1

    if own["counter"] == 5:
        assignFacesRandom('spin', 0, 180, 0.0, 20.0, True)

        try:
            count = open("D:/praace/pokusy/rainbow/counter.txt", "w")
        except:
            print ("counter.txt ...bussy" + "\n")
        count.write(str(own["counter"]+1))
        count.close()

        own["restart"] = 1

    if own["counter"] == 6:
        c = random.randint(0, len(paleta)-1)
        transformFaces('spin', spin_Amplitude, spin_Len, -0.2, spin_StepAngle,
spin_Phase, spin_Step, spin_ModAmplitude, spin_ModStep, [-2.22045e-16, -1, -0],
```

```
spin_Joint, c)

        try:
            count = open("D:/praace/pokusy/rainbow/counter.txt", "w")
        except:
            print ("counter.txt ...bussy" + "\n")
        count.write(str(own["counter"]+1))
        count.close()

        own["restart"] = 1

    if own["counter"] == 7:
        if balls == 1:
            print("balls True")
            assignFacesBall("ball")
            addBalls("ball", ball_Size, ball_Hairy, rand_Hairy, 3)
        else:
            print("no balls")
        try:
            count = open("D:/praace/pokusy/rainbow/counter.txt", "w")
        except:
            print ("counter.txt ...bussy" + "\n")
        count.write(str(own["counter"]+1))
        count.close()

        own["restart"] = 1

    if own["counter"] == 8:
        try:
            count = open("D:/praace/pokusy/rainbow/counter.txt", "w")
        except:
            print ("counter.txt ...bussy" + "\n")
        count.write(str(own["counter"]+1))
        count.close()

        own["restart"] = 1

    if own["counter"] == 9:
        bpy.ops.object.modifier_add(type='SUBSURF')
        bpy.data.objects["Plane"].modifiers["Subsurf"].levels = 2

        try:
            count = open("D:/praace/pokusy/rainbow/counter.txt", "w")
        except:
            print ("counter.txt ...bussy" + "\n")
        count.write(str(own["counter"]+1))
        count.close()

        try:
            count = open("D:/praace/pokusy/rainbow/var.txt", "w")
        except:
            print ("var.txt ...bussy" + "\n")
        count.write("1")
        count.close()

        own["restart"] = 1

    bpy.ops.object.mode_set(mode='OBJECT')
    bpy.ops.object.shade_smooth()
    print ("generate done")
execute()
```

//python program v prostředí Blenderu – opakovaný start

Modul „hlídá“ proměnnou v textovém dokumentu,aby mohl začít nový cyklus běhu programu.

```
cont = bge.logic.getCurrentController()
own = cont.owner

try:
    goal = open("D:/praace/pokusy/rainbow/start.txt", "r")
#    print ("start.txt opened")
except:
    print ("start.txt ...bussy" + "\n")

own["start"] = int(goal.read())
goal.close()

if own["start"] == 1 and own["counter"] = 0:
    try:
        goal = open("D:/praace/pokusy/rainbow/start.txt", "w")
        print ("start.txt opened")
    except:
        print ("start.txt ...bussy" + "\n")

    goal.write("0")
    goal.close()
```

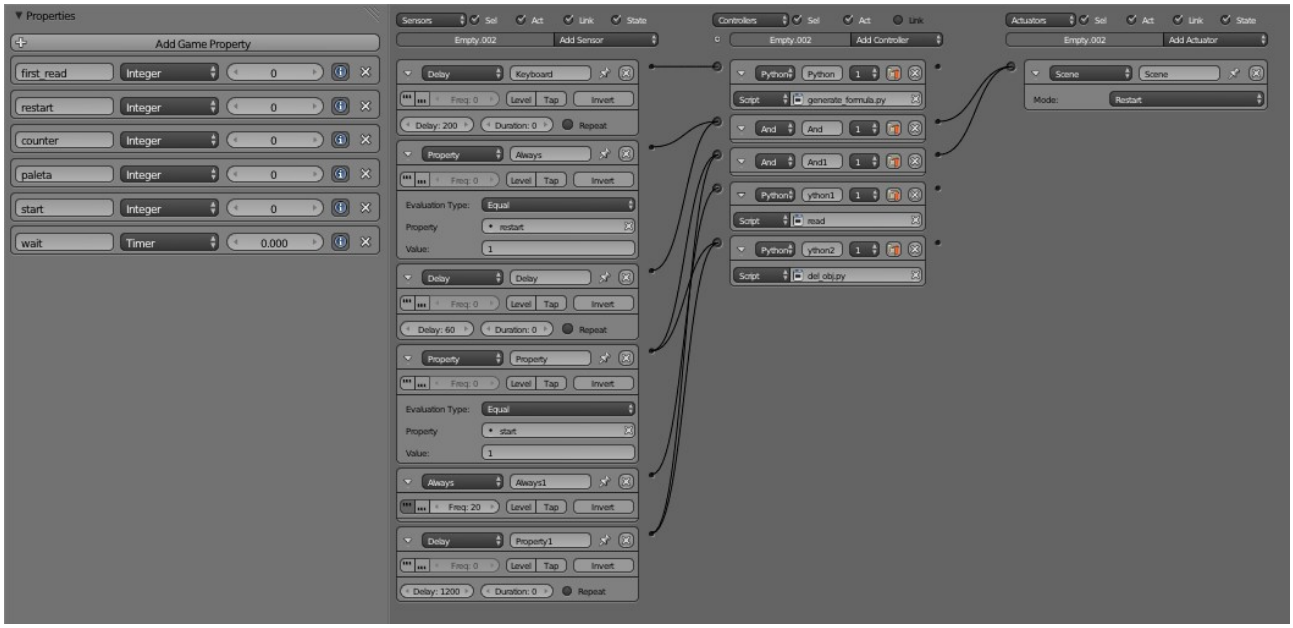
//python program v prostředí Blenderu – vymazání objektu

Aby nedocházelo k nežádoucí muhromadění objektů, jsou po uplynutí určité doby ze scény odstraněny.

```
import bpy

bpy.ops.object.delete()
print("object deleted")
```

//schéma zapojení logických prvků v Blenderu (BGE)



Já sám jsem technicky zaměřený a mým cílem je vytváření umění s použitím moderní techniky. Věda a technika šly vždy ruku v ruce a bylo by škoda je od sebe rozdělovat. Věda a technika obohacují umění a umění inspiruje techniku a pokrok.

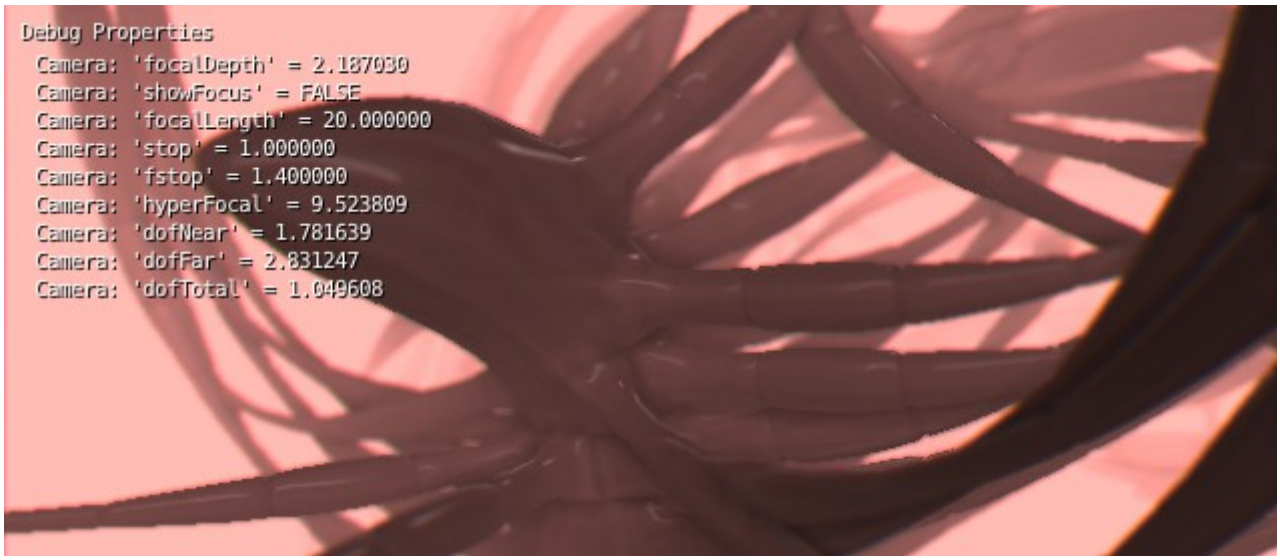
Touto prací chci divákovi odkrýt krásu běžných praktických věcí, která bývá méně vnímavým lidem skryta.

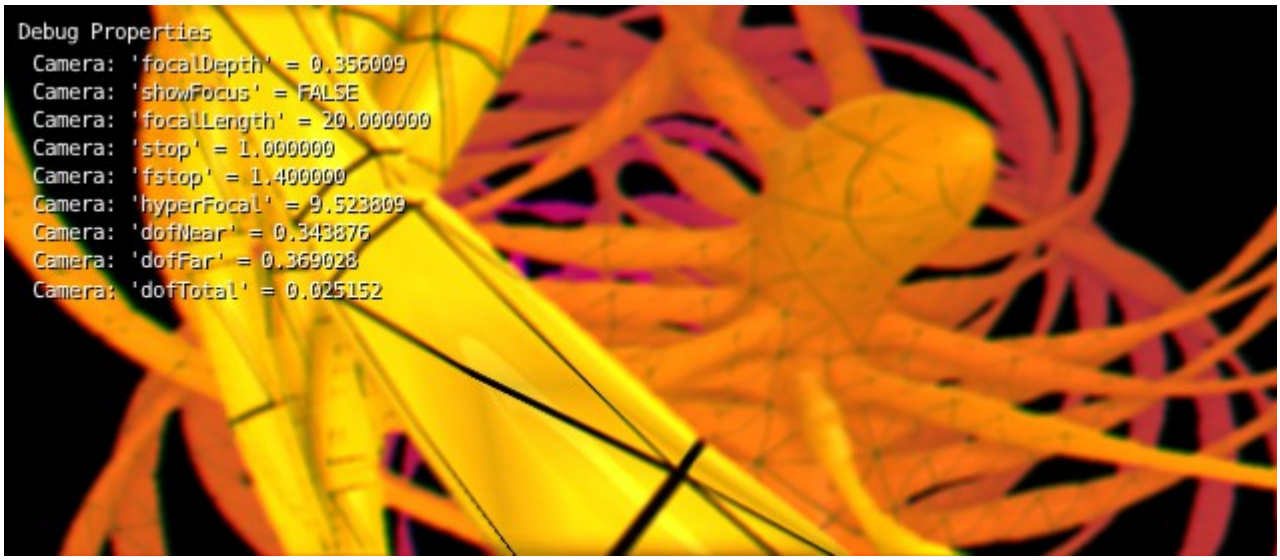
Tiskárna svým procesem zároveň poukazuje na informace, které jsou všude kolem nás, a které si ani neuvědomujeme a převádí je do srozumitelné formy pro širokou veřejnost. Z abstraktního elektrosmogu se tak stává konkrétní umělecké dílo.

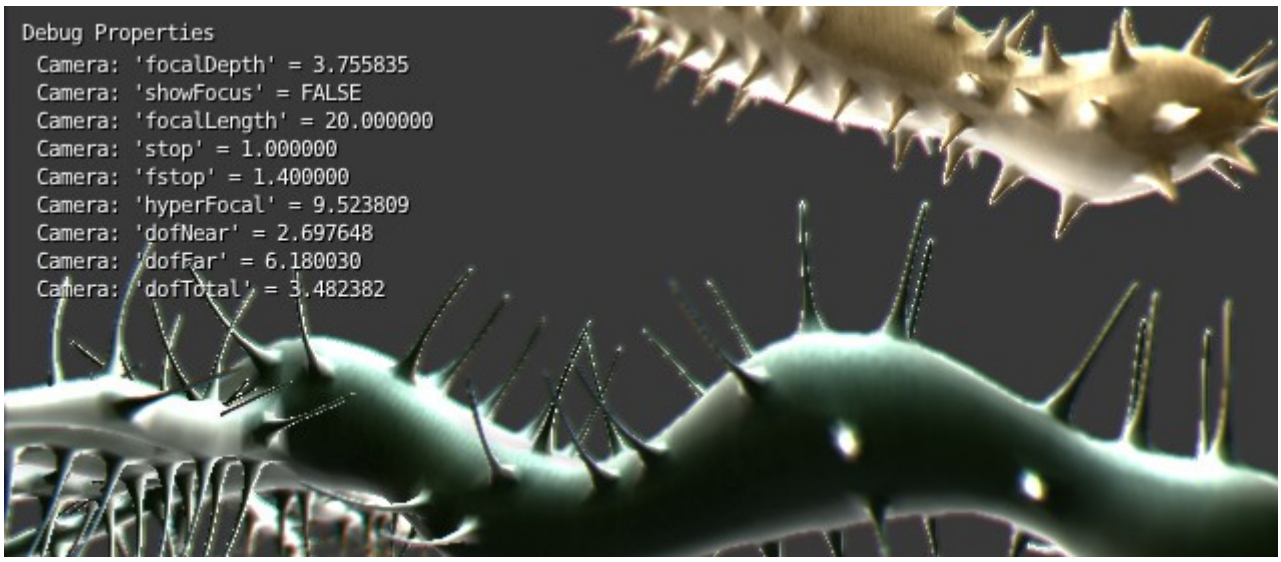
Děkuji panu Doc. Stanislavu Zippemu za cenné rady, které mě přivedly až k diplomové práci, Mgr. Jaroslavu Prokešovi za podporu a v neposlední řadě Ing. Richardovi Charvátovi za odborné konzultace.

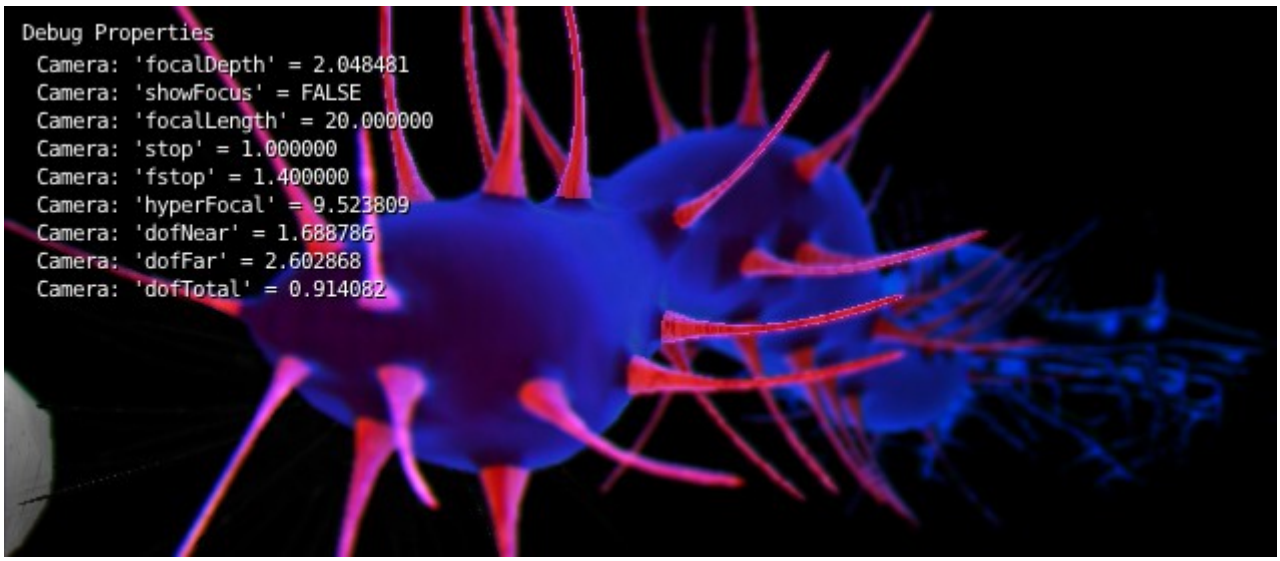
prostorová instalace

vizualizace objektů








Debug Properties
Camera: 'focalDepth' = 2.048481
Camera: 'showFocus' = FALSE
Camera: 'focalLength' = 20.000000
Camera: 'stop' = 1.000000
Camera: 'fstop' = 1.400000
Camera: 'hyperFocal' = 9.523809
Camera: 'dofNear' = 1.688786
Camera: 'dofFar' = 2.602868
Camera: 'dofTotal' = 0.914082


Debug Properties
Camera: 'focalDepth' = 1.593284
Camera: 'showFocus' = FALSE
Camera: 'focalLength' = 20.000000
Camera: 'stop' = 1.000000
Camera: 'fstop' = 1.400000
Camera: 'hyperFocal' = 9.523809
Camera: 'dofNear' = 1.367397
Camera: 'dofFar' = 1.908570
Camera: 'dofTotal' = 0.541173

3D objekt