OmegaL

The Cake4D Script Language created by **TAP OK G**
The text edited by **Strokovblack**
Cake4D version 0.1.4 KS

www.tapokg.org
All rights reserved.
Italy 2022

Script Language

Main Methods

OmegaL script is always 1 file that can be automatically constructed from several files by **import** feature.

Script file must have 2 **script methods**:
init,
update

```
int a;
int b;
init{
    a = 0xD;    //0000 1101 = 13
    b = 0x18;   //0001 1000 = 24
    a|b;
    printInt(); //0001 1101
    printString("\t= 29\n");
}

update{
    quit();
}
```

Script method **init** - calls only once and should be used to:
- Init all textures
- Init system calls
- Init all objects

Init calls before render cycle, so if **object** is null after **init** method it will not appear in the game.

Script method **update** calls each render call before OpenGL draw call and should be used for:
- Objects updates
- Game mechanics updates
- UI update

Script Methods

You can create the script methods anywhere in the script files and call them from any point of the file.

```
init{
	method_1(); //this will create endless cycle of redirecting
}

method_1{
	method_2();
}

method_2{
	method_1();
}
```

Import

You can import methods from any scripts files:

mods/SC_1.oml:

```
SC_1_do_math{
	a|b;
	printInt(); //0001 1101
	printString("\t= 29\n");
}
```

mods/bittest.omg4d:

```
import "mods/SC_1.oml";

int a;
int b;
init{
	a = 0xD;	//0000 1101 = 13
	b = 0x18;	//0001 1000 = 24
	SC_1_do_math();
}

update{
	quit();
}
```

Result script file will be like this:

```
SC_1_do_math{
	a|b;
	printInt(); //0001 1101
	printString("\t= 29\n");
}

int a;
int b;
init{
	a = 0xD;	//0000 1101 = 13
	b = 0x18;	//0001 1000 = 24
	SC_1_do_math();
}

update{
	quit();
}
```

This way you do not need to rewrite the same code for each new script file, you can just import it.

Override

You can also override methods from imported files if you need to change something.

mods/bittest.omg4d:

```
import "mods/SC_1.oml";

SC_1_do_math{
	a&b;
	printInt(); //0000 1000
	printString("\t= 8\n");
}

int a;
int b;
init{
	a = 0xD;	//0000 1101 = 13
	b = 0x18;	//0001 1000 = 24
	SC_1_do_math();
}

update{
	quit();
}
```

Result file will look like this:

```
SC_1_do_math{
	a&b;
	printInt(); //0000 1000
	printString("\t= 8\n");
}

int a;
int b;
init{
	a = 0xD;	//0000 1101 = 13
	b = 0x18;	//0001 1000 = 24
	SC_1_do_math();
}

update{
	quit();
}
```

If you have the same name methods inside different imports the final one will be from the first import.

```
import “mods/SC_1.txt”;// printString(“SC_1\n"); <- from this one
import “mods/SC_2.txt”;// printString("SC_2\n");
import “mods/SC_3.txt”;// printString("SC_3\n");

init{
    SC_print(); // -> SC_1
}

update{
    quit();
}
```

If you have imports in imports files the load order will be form this way:

```
SC_1.txt
      SC_3.txt
            SC_3_1.txt
            SC_3_2.txt
      SC_1_1.txt
            SC_1_1_1.txt
      SC_1_2.txt

SC_2.txt
      SC_2_1.txt
      SC_2_2.txt

SC_3.txt
      SC_3_1.txt
      SC_3_2.txt
```

Result order:

```
SC_1.txt
SC_2.txt
SC_3.txt
~~SC_3.txt~~ // will be ignored because was already loaded
SC_1_1.txt
SC_1_2.txt
SC_2_1.txt
SC_2_2.txt
SC_3_1.txt
SC_3_2.txt
SC_1_1_1.txt
```

You can use any variables and any methods from all imports because the result script will contain all parts of all imports, except the override methods.

Classes

This is a full list of all classes which exist in the OmegaL:

```
//static classes (CAN ONLY BE INSIDE SCRIPT METHODS)
-4:    b
-3:    s
-2:    i
-1:    f

//numbers (LINK SAVE)
0 :  float
1 :    int
16:   bool

//OBJECTS (LINK DANGEROUS)

//String objects
15: string

//Vector objects
2 :   vec2
3 :   vec3
4 :   vec4

//Cake4D objects
6 :  lit4d  // light object
5 :  cam4d  // used to keep different camera parameters
7 :  obj4d  // used to draw static model
10:  sky4d  // used to draw background (extended obj4d)
8 : aobj4d  // used to draw animated model (extended obj4d)
9 : pers4d  // used to draw mob (extended aobj4d)
30: part4d  // used to draw primitive 3D particles inside 4D

// 2D objects
11:  img2d  // used to draw 2D texture (for example UI)
13: text2d  // used to draw 2D text

//currently useless
12:  muc4d

//script function class (can be used inside code directly)
14: method

//USES TO RECIVE NUMBER COMPONENTS OF OBJECTS (float)
17:   vec2.x
18:   vec2.y
19:   vec3.x
20:   vec3.y
21:   vec3.z
22:   vec4.x
23:   vec4.y
24:   vec4.z
25:   vec4.k
26: aobj4d.animtime //animation time
27: pers4d.animtime //animation time
```

Static Classes

static boolean

```
init{
	//bool - system variable
	bool = true;
	printBool(); //true
	printString("\t= true\n");

	bool = false;
	printBool(); //false
	printString("\t= false\n”);

	printBool(true); // compiler turns it into 2 lines
	                 // bool = true; printBool();
	printString("\t= true\n");

}
```

static float

```
init{
	//float - system variable
	float = 1.0;
	printFloat(); //1.0
	printString("\t= 1.0\n");

	float =-0.989;
	printFloat(); //0.989
	printString("\t= -0.989\n”);

	printFloat(1.0); // compiler turns it into 2 lines
	printString("\t= 1.0\n”);

	printFloat(-1.0); printString(“\t= -1.0\n”);

}
```

static int

```
init{
	//int - system variable
	int = 555;
	printInt();    printString("\t= 555\n0x”);
	printIntHex(); printString("\t= 0x22b\n");
	printIntOct(); printString("\t= 1053\n");
	printIntBin(); printString("\t= 1000101011\n”);

	int = 0xFFA522;
	printInt();    printString("\t= 16753954\n0x”);
	printIntHex(); printString("\t= 0xffa522\n”);

	int = -555;
```

```
	printInt(); printString("\t= -555\n”);

	printInt( 555); printString("\t=  555\n”);
	printInt(-555); printString("\t= -555\n”);
}
```

static string

```
init{
	//string - system variable
	string = “\t "; printString(); // tab
	string = “\n "; printString(); // new line
	string = “Hello, World!\n"; printString();
}
```

links

Link is actually a **static int** that keeps its line number inside its **script method.**

It is important to know because as any other **int number** they use **int** in **if( )** method.

```
init{
	float = 0.0;
	method_1();
	method_2();
	print();
	// output:
	// int	float	bool	method
	//  0	- 0.0	- false	method_1
	// -1	- 1.0	- true	method_1
	// -1	- 1.0	- false	method_2
	//  0	- 2.0	- true	method_2
	//  0	- 3.0	- false	init
}

method_1{
	LINK : link_again;// = -1
		print();
		float += 1.0;
		bool = float < 1.1;
	if(bool, link_again); // equals if(bool, -1);
}

method_2{
	float -= 1.0;
	LINK : link_again;// = 0
		print();
		float += 1.0;
		bool = float < 2.1;
	if(bool, link_again);  // equals if(bool, 0);
}
```

```
print{
      printInt();   printString("\t - ");
      printFloat(); printString("\t - ");
      printBool();  printString("\n");
}
```

Numbers

Numbers have names and hence can be used everywhere in the program, so it's very important to keep the right name for variable.

```
bool  var_bool;
int   var_int;
float var_float;

init{
      var_bool = true;
      printBool(var_bool);   printString("\t= true\n");

      var_int = 444;
      printInt(var_int);     printString("\t= 444\n”);

      var_float = 4.44;
      printFloat(var_float); printString("\t= 4.44\n”);

      //Transforms to:

      var_bool = true;
      bool =  var_bool;           printBool();
      string  =  "\t= true\n";    printString();

      var_int = 444;
      int = var_int;              printInt();
      string  =  "\t= 444\n”;     printString();

      var_float = 4.44;
      float  = var_float;         printFloat();
      string  =  "\t= 4.44\n";    printString();

      //system variables that have names of their classes
      //  are used to transfer data to the system functions
}
```

Primitive Objects

string objects

```
string var_str;
init{
      //string objects are safe to #= operations
      //string = var_str; string #= var_str; -> string = var_str;
      var_str = “Hello, World!\n";
      printString(var_str);
}
```

vector objects

```
vec2 vec_2;
vec3 vec_3;
vec4 vec_4;
vec2 vec_2_bad;

init{
      //vec_2(0.0); equals vec2(0.0); vec_2 #= vec2;
      vec_2(0.0); vec2.x = 0.01; vec2.y = 0.02;
      vec_3(0.0); vec3.x = 0.01; vec3.y = 0.02; vec3.z = 0.03;
      vec_4(0.0); vec4.x = 0.01; vec4.y = 0.02;
                  vec4.z = 0.03; vec4.k = 0.04;

      printVec2(); printString(“\t= vec_2 \n");
      printVec3(); printString("\t= vec_3 \n");
      printVec4(); printString("\t= vec_4 \n\n");

      //vector objects are not safe to link operations
      //you need to keep in mind which object
      //  currently binds to vec2, vec3, vec4

      vec_2_bad(5.0);
      //why bad?
      printVec2(vec_2_bad); printString("\t= vec_2_bad \n");
      printVec2(vec_2);     printString("\t= vec_2 \n");
      printVec2(vec_2_bad); printString("\t= vec_2_bad \n\n”);

      //output:
      //
      //5.00, 5.00     = vec_2_bad
      //0.01, 0.02     = vec_2
      //0.01, 0.02     = vec_2_bad
      //

      //compiler code:
      float = 5.0; vec_2_bad #= vec2(float);
      //after this line vec2 & vec_2_bad link to the same data
```

```
        //this equals vec_2_bad = vec_2_bad;
        vec2 = vec_2_bad; printVec2(); string = "\t= vec_2_bad \n";
        printString();
        //this equals vec_2_bad = vec_2;
        vec2 = vec_2;     printVec2(); string = "\t= vec_2 \n";
        printString();
        //this equals vec_2_bad = vec_2_bad;
        vec2 = vec_2_bad; printVec2(); string = "\t= vec_2_bad \n\n";
        printString();

}
```

This happens because variables of **vec2, vec3, vec4** and other **objects** are actually **links** to the data and are not the actual data like in case of **numbers**.

Light Objects

lit4d object

Lit4d is **object** that keeps light data in **Cake4D**.

First 10 **lit4d** objects that are in the script will be used to draw scene. Extra light data will be ignored.

Theoretically you can create any amount of lit4d but only first 10 links will be actually used to draw scene, but you can use #= operator to manage this.

```
      lit4d sun_light;
      vec3 lights_strength_on;
      vec3 lights_strength_off;
init{

      lit4d();
      lit4d.position;       vec4.y = 10.0;
      lit4d.attenuation;    vec3.x = 1.0; vec3.y = 0.0;vec3.z = 0.0;
      lit4d.strength;       vec3.x = 0.8; vec3.y = 0.3;vec3.z = 0.2;
      lit4d.color;          vec3.x = 1.0; vec3.y = 0.7;vec3.z = 0.5;

      sun_light #= lit4d;
      sun_light.position;   printVec4();printString("\t= position \n");
      sun_light.attenuation;printVec3();printString("\t= attenuation\n");
      sun_light.strength;   printVec3();printString("\t= strength \n");
      sun_light.color;      printVec3();printString("\t= color \n");

      lights_strength_on( 1.0);
      lights_strength_off(0.0);

      lights_strength_on = sun_light.strength;

      //dispose
      vec3 #= null;
}

light_on{
      sun_light.strength; vec3 = lights_strength_on;
      printVec3();      printString("\t= strength ON \n");
      //dispose
      vec3 #= null;
}

light_off{
      sun_light.strength; vec3 = lights_strength_off;
      printVec3();      printString("\t= strength OFF\n");
      //dispose
      vec3 #= null;
}

update{
      light_on();
```

```
      light_off();
      quit();
}
```

The equation to calculate light in Cake4D:

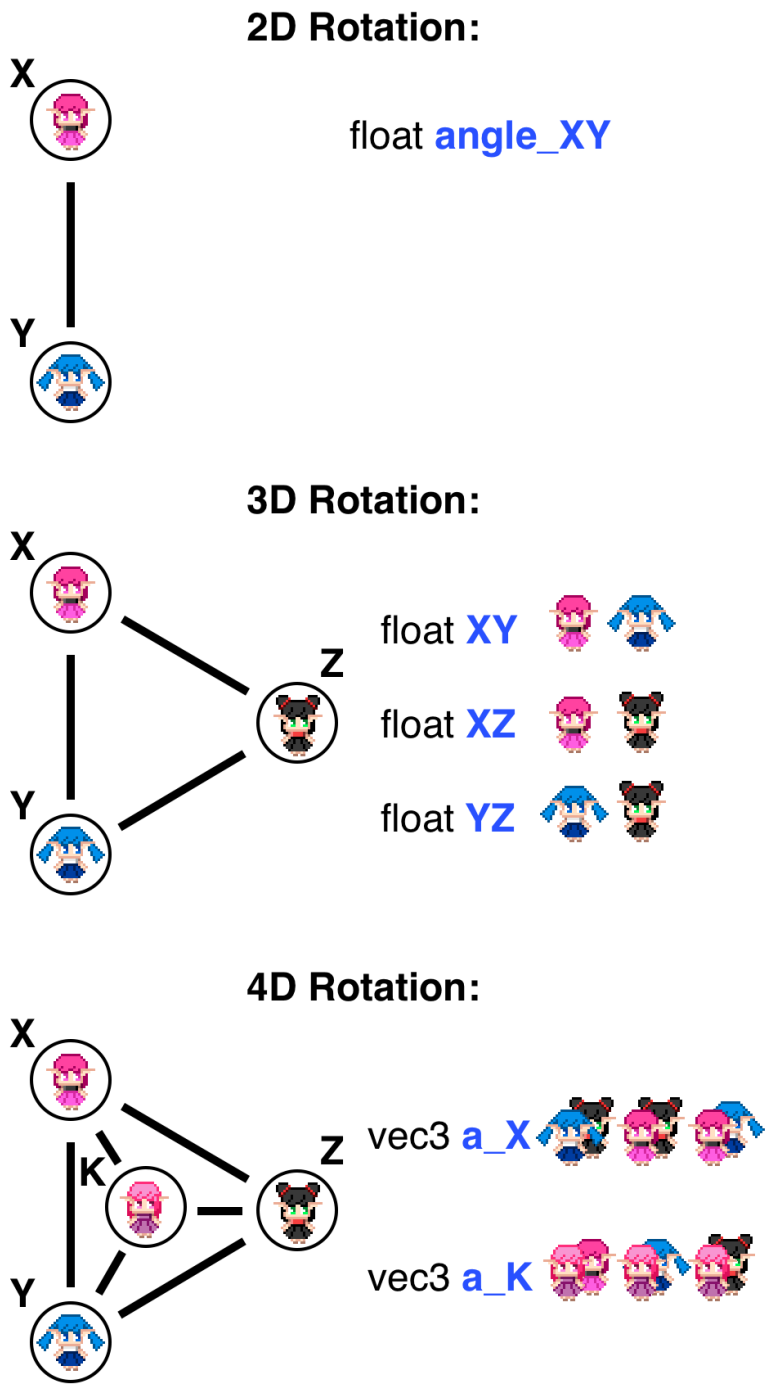
```
vec3 A - lit4d.attenuation;
vec3 S - lit4d.strength;
vec3 C - lit4d.color;

float d - distance between lit4d and object4d;

attenuation = 1.0/(A.x + A.y*d + A.z*d*d);

result = C*(ambient*S.x + diffuse*S.y + specular*S.z)*attenuation
```

4D Rotation

About 4D:

There is a very simple way to understand rotation in general.

Like *a couple* **Rotation** is **only possible between two**.



cam4d, obj4d, sky4d, aobj4d, pers4d - have these angles.

4D Objects

cam4d object

```
cam4d cam;

init{
	//init
	cam4d();   cam #= cam4d;
	//set cam active
	setCamera();

	//Camera parameters IN
	cam.position;  //vec4(x, y, z, k)
	cam.angleK;    //vec3(KX, KY, KZ)
	cam.angleX;    //vec3(YZ, XZ, XY)

	//need to be called to calculate rotation
	cam.update();

	//View directions OUT
	cam.move_K; //vec4 to cam4d forward
	cam.move_X; //vec4 to cam4d left
	cam.move_Z; //vec4 to cam4d right

	//Dispose
	vec4 #= null;
	vec3 #= null;
}
```

obj4d object

In order to easy init all parameters of all **obj4d, aobj4d, lit4d, part4d** you can use

Map 4D Editor.

```
import “maps_projects/levels/map/SC.map”;//keeps method obj4d_map_init()

init{
	obj4d_map_init();
}
```

Or you can do it manually:

```
obj4d my_obj;
init{
	//model file name
	string = “models/textured_wall.json";
	obj4d(string);//init;
	//obj4d
	obj4d.angleK	;vec3.x=0.0; vec3.y=-1.5; vec3.z= 0.0;
	obj4d.angleX	;vec3.x=0.0; vec3.y=-0.0; vec3.z= 0.0;
	obj4d.position	;vec4.x=0.0; vec4.y=-0.0; vec4.z= 0.0; vec4.k=0.0;
	obj4d.scale	;vec4.x=6.0; vec4.y= 6.0; vec4.z= 6.0; vec4.k=3.0;

	//Material (ambient&diffuse, specular, specular_form)
	obj4d.color	;vec3.x=1.0; vec3.y= 1.0; vec3.z= 1.0;
	//Step Sound
	obj4d.setSound(2);

	//Texture
	obj4d.tex_offset;vec3.x=0.0; vec3.y = 3.0;  vec3.z= 0.0;
	obj4d.tex_size  ;vec3.x=1.0; vec3.y = 1.0;  vec3.z= 1.0;
	obj4d.tex_mult  ;vec3.x=6.0; vec3.y = 6.0;  vec3.z= 6.0;

	//Rigid. it is not necessary to draw object 4D
	obj4d.createRigid();
	obj4d.rigidSize  ; vec4.x=1.0; vec4.y=1.0; vec4.z=1.0; vec4.k=0.05;
	obj4d.rigidCenter; vec4.x=0.0; vec4.y=0.0; vec4.z=0.0; vec4.k=0.0;
	// (0000 0000) there are 8 rigid cubes around each obj4d
	// 128 - means that only the 1st cube is on. (1000 0000 = 128)
	obj4d.rigidInitPoints(128); //used to set rigid cubs

	//Update rotation matrix and rigid
	obj4d.update();

	//Make link
	my_obj #= obj4d;
}

update{
	//it changes visibility
	my_obj.setVisible(false);
	//returns visibility state
	my_obj.isVisible(); printBool(); printString(“\t= is Visible\n”);

	//returns drawing ID
	//works only in update cycle
	my_obj.getID(); printInt(); printString("\t= Draw ID\n”);

	//returns sound ID, can be used to play its sound
	my_obj.getSound();  printInt(); printString("\t= Sound ID\n”);

	//can be used to turn off/on rigid
	my_obj.setRigidCheck(false);

	quit();
}
```

sky4d object

Sky4d extends obj4d and has the same methods and objects.

This is the code part from “modules/LVL.oml”

```
    sky4d  LVL_sky;
    string LVL_sky_filename;
    int    LVL_sky_size;
LVL_sky_init{
      //setting sky texture parameters
      int = LVL_sky_size;
      // size x size x size - 1 cube (sky has 8 of them)
      // texture should have 2D size as (size*size) x (size*8)
      // sky_64.png (4096x512) has size 64
      setSkyTexture(LVL_sky_filename);
      //sky model name
      string = "models/sky4d.json";
      //init sky4d object
      LVL_sky = sky4d(string);
      LVL_sky.scale; // vec4 #= sky4d.scale;
      getFOV_FAR();  // this provides camera vision distance
      float *= 0.5;
      vec4 *= float; // set sky size to match camera vision ability
      vec4 #= vec4_null;


}

update{
      //returns drawing ID
      //works only in update cycle
      LVL_sky.getID(); printInt(); printString("\t= Draw ID\n”);
}
```

aobj4d object

```
aobj4d cake4d;
init{
      //animated 4D model
      string = "models/cake.ajson";
      //cake_4D
      aobj4d(string);
      //obj4d extended parameters
      aobj4d.angleK    ;//vec3
      aobj4d.angleX    ;//vec3
      aobj4d.position  ;//vec4
      aobj4d.scale     ;//vec4

      //Material (ambient&diffuse, specular, specular_form)
      aobj4d.color     ;//vec3
      //Texture
```

```
      aobj4d.tex_offset;//vec3
      aobj4d.tex_size  ;//vec3
      aobj4d.tex_mult  ;//vec3

      //update rotation matrix
      aobj4d.update();


      //set time of animation
      aobj4d.animtime += 1.0;
      //calculate animation (without first calculation aobj won't appear)
      aobj4d.updateA();
      //set visibility
      aobj4d.setVisible(false);
      //set link
      cake4d #= aobj4d;
}

update{
      //returns drawing ID
      //works only in update cycle
      aobj4d.getID(); printInt(); printString("\t= Draw ID\n");
}
```

pers4d object

To explain how pers4d object should be used it will be easer to just show a simplified example of the game code from "modules/LVL.oml"

```
import "modules/KEY.oml";

      vec4   LVL_vec4_S;
      pers4d pers_Alla;
      cam4d  Alla_pov;
LVL_pers4d_init{
      //new pers4d
      pers4d("models/alla.ajson");
      //Set height
      // girl_2 model height is 205.600 cm ( 1m = 1.0)
      // Alla height is 164 cm = 1.64 m
      float = 164.0 / 205.6; // model scale
      // the use of named variable is necessary because .scale takes vec4
      LVL_vec4_S(float); pers4d.scale = LVL_vec4_S;
      // default physical parameters
      vec4(0.0); //clear vec4

      vec4.x = 58.00; // x - mass                  kg
      vec4.y = 1.890; // y - walking_speed         m/s
      vec4.z = 2.230; // z - running_speed         m/s
      vec4.k = 5.310; // k - super_running_speed   m/s
      pers4d.setPhysicalParameters();

      vec4.x = 11.11;       // x - energy time cost              cal/s
      vec4.y = 34.47;       // y - starting running cost         cal/s
```

```
      vec4.z = 1.640;       // z - height                          m
      vec4.k = 38396000.0;  // k - Maximum Calorie capacity        cal
      pers4d.setEnergyParameters();

      vec4.x = 300.0; // x - jump focus cost                 cal/s
      vec4.y = 200.0; // y - jump min energy                 cal
      vec4.z = 300.0; // z - jump max energy                 cal
      vec4.k =   0.0; // k - currently unused
      pers4d.setJumpParameters();

      //  Cake4d 1/6 piece (125 g) - 440 kcal
      //  Full Cake4d - 2,640 kcal
      //
      //  Calorie per days
      //  Alla  - 2,005 kcal/day
      //  Ren   - 2,000 kcal/day
      //
      // should be lower then the maximum calorie capacity
      pers4d.setEnergy(38395000.0);
      // set link
      pers_Alla #= pers4d;

      // create camera
      cam4d(); Alla_pov #= cam4d;
      // attach to Alla
      pers_Alla.setCamera();

      // set camera distance from pers4d
      pers_Alla.setCameraRadius(2.0);

      // set camera height
      pers_Alla.camPosition; vec4.y = 1.64 * 0.85;
      // set model angle
      pers_Alla.angleK; vec3.x = 0.5;
      // update rotation matrix and the camera
      pers_Alla.update();

      // update animated model
      pers_Alla.animtime = 0.0;
      pers_Alla.updateA();
}

vec4 LVL_vec4_move;
Vec4 LVL_vec4_1;
LVL_pers4d_update{
      //getKey() - system function
      //key_W, key_jump, key_Shift - from “modules/KEY.oml"

      //Jump
      getKey(key_jump);                 //bool - try to jump
      pers_Alla.jump(bool);

      //Move
      LVL_vec4_1(0.0);
      getKey(key_W); !bool; if (link_K_W);
            LVL_vec4_move = Alla_pov.move_K; //camera direction
```

```
            LVL_vec4_1 += LVL_vec4_move;
        LINK : link_K_W;

        vec4(0.0); vec4 = LVL_vec4_1;    //vec4 - move direction
        getKey(key_Shift);               //bool - try to speed up
        pers_Alla.move(vec4, bool);


        // update gravity
        pers_Alla.updateGravity();
        // check all collisions
        pers_Alla.checkCollision();
        // updates rotation matrix and camera
        pers_Alla.update();

        //returns drawing ID
        //works only in update cycle
        pers_Alla.getID(); printInt(); printString("\t= Draw ID\n");
}
```

part4d object

```
        part4d my_part4d;
init{
        part4d();

        // general
        part4d.position  ; //vec4 - position
        part4d.scale     ; //vec4 - size
        part4d.color     ; //vec3 - material

        // texture
        part4d.tex_offset; //vec3 - texture part position
        part4d.tex_size  ; //vec3 - texture part size
        part4d.tex_mult  ; //vec3 - texture part repeating

        // visibility
        part4d.setVisible(false);
        // link
        my_part4d #= part4d;
}

update{
        //returns drawing ID
        //works only in update cycle
        part4d.getID(); printInt(); printString("\t= Draw ID\n");
}
```

2D Objects

img2d object

```
img2d cakeBar;
init{
	//load 2D texture to draw img2d from "imgs/" folder
	loadTex2d("ui_texture.png");

	cakeBar = img2d();
	cakeBar.setMarginX(1);//0-left. 1-center, 2-right
	cakeBar.setMarginY(1);//0-bottom. 1-center, 2-top
	cakeBar.psize;		//size and position on screen
		vec4.x =  1.00; //x
		vec4.y =  0.88; //y
		vec4.z =  0.20; //width
		vec4.k =  0.20; //height

	//dxwh - x distance that changes with screen height
	//x += dxwh*screen_w/screen_h
	float = -0.1; cakeBar.setDXHW();

	cakeBar.tex_psize;	//size and position on texture
		vec4.x = 0.5;	//x
		vec4.y = 0.1;	//y
		vec4.z = 0.1;	//width
		vec4.k = 0.1;	//height

	cakeBar.color;		//vec4 result = texture_color * color;
		vec4.x = 1.0;	// - level of R channel
		vec4.y = 1.0;	// - level of G channel
		vec4.z = 1.0;	// - level of B channel
		vec4.k = 0.5;	// - level of A channel

	cakeBar.affine;		//Affine matrix (can be used to rotate img2D)
		vec4.x = 1.0;	vec4.y = 0.0;	// x' = a.x*x + a.y*y;
		vec4.z = 0.0;	vec4.k = 1.0;	// y' = a.z*x + a.k*y;

	//Visibility
	cakeBar.setVisible(true);
	cakeBar.isVisible(); //bool
}
```

text2d object

```
text2d Cake4D;
text2d Cake4D_2;
init{
	//loads text from "text/"+lang+"/launcher.json"
	// default lang is "en"
```

```
// you can add any lang text and select it in launcher
//         new text folder must match en version and
//         new lang options must be added to "text/lang_list.txt"
// you can load several text files or all of them
loadText("launcher.json");

      int = 0;         //Text ID from "launcher.json"
      vec3(0.0);
      vec3.x =  0.0;   //x on screen
      vec3.y =  0.5;   //y on screen
      vec3.z =  0.7;   //text size
float = 1.0;     //text animation 0.0 - 0 characters, 1.0 - all

Cake4D = text2d();

// reset Text ID from "launcher.json"
Cake4D.setTextID(0);

// align text to the left
Cake4D.needCenter(false);
// align text to the center
Cake4D.needCenter(true);

//dxwh
float = -0.80; Cake4D.setDXHW();

//text appearing animation [0.0 - 1.0]
Cake4D.setTextAnimation(1.0);

//vec4 the same as with text2d
Cake4D.color;

//Visibility
Cake4D.setVisible(true);
Cake4D.isVisible(); //bool


int = 0;
vec3(0.0); vec3.x =  0.0; vec3.y = -0.5; vec3.z = 0.7; float = 1.0;
Cake4D_2 = text2d();

//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
//Next operations affect all text2d objects with current 0 Text ID
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Cake4D.resetOldSize("new Cake4D”);
//result: "new Ca" - old size: 6
Cake4D.resetNewSize("new Cake4D”);
//result: "new Cake4D" - new size: 10

//the fastest operation but
//    doesn't change size and
//    works only with ENG symbols
Cake4D.resetEng("new new Cake4D");
//result: "new new Ca" - old size: 10
```

```
        // COLORS
        vec3(1.0);
        Cake4D.setFColor();// vec3 font color

        // you have to init new vec3, because old one is linked to vec3
        vec3(1.0); vec3.y = 0.0; vec3.z = 0.0;
        Cake4D.setSColor();// vec3 shade color

        //clear
        vec3(0.0);
        //result: font: WHITE, shade: RED

        //TEXT Edit
        Cake4D.textEdit_clear(); // set size 0

        //add character by its id
        Cake4D.textEdit_add(35); // C
        Cake4D.textEdit_add(33); // A
        Cake4D.textEdit_add(43); // K
        Cake4D.textEdit_add(37); // E
        Cake4D.textEdit_add(20); // 4
        Cake4D.textEdit_add(36); // D
        Cake4D.textEdit_add(36); // D

        //delete last character
        Cake4D.textEdit_delete();

        //print text to console
        Cake4D.textEdit_print();
}
```

Class operators

Operator =

Works with any classes

```
	vec3 a_vec3;
	vec3 b_vec3;

init{

	float = 0.0; // float = 0.0;
	float = 0;   // float = 0.0;

	int   = 0;    // int = 0;
	int   = 0.5;  // int = 0;
	int   = false;// int = 0;
	int   = true; // int = 1;

	a_vec3(1.0);

	b_vec3 = a_vec3; // b_vec3(0.0); b_vec3 = a_vec3;

	a_vec3.x = 2.0; vec3(0.0);

	//a_vec3 is not b_vec3
	printVec3(a_vec3); //2,1,1
	printVec3(b_vec3); //1,1,1

	// = operator only copies the data of objects,
	// it doesn't change links
	// this operator works with any object
}
```

Operator #=

Works with **objects**

Sets data link

```
	vec3 a_vec3;
	vec3 b_vec3;
init{
	a_vec3(1.0); // data A has been created, a_vec3 - link, vec3 - link
	b_vec3(2.0); // data B has been created, b_vec3 - link, vec3 - link

	printVec3(); printString(" - 2.0 \n"); //data B (link vec3)
	vec3   #= a_vec3; // link vec3 = link a_vec3 (data A)
	printVec3(); printString(" - 1.0 \n"); //data A (link vec3)
	a_vec3 #= null;   // link a_vec3 disposed
	printVec3(); printString(" - 1.0 \n"); //data A (link vec3)
	vec3   #= null;   // link vec3 disposed
```

```
	printVec3(); printString(" - null \n"); //no data (link vec3)
}
```

Operators += and -=

Works only with **numbers**(not bool), **vectors** and **strings**(only +=)

```
	vec4 a_vec4;
init{
	int = 1;

	int += 2;		// int = 3
	int -= 5;		// int = -2
	printInt();

	float = 5.0;
	float += 4;		// float = 9.0
	float -= int;		// float = 9.0 - (-2) = 11.0
	printFloat();

	string = "Cake";
	string += "4D";
	string += "\n";
	printString(); // -= doesn't work

	a_vec4(1.0);
	vec4(2.0);

	vec4 += a_vec4;
	printVec4(); // 3,3,3

	vec4(0.0);
	vec4 -= a_vec4;
	printVec4(); // -1,-1,-1
}
```

Operators *= and /=

Works only with **numbers**(not bool) and **vectors**

```
init{
	string = "\n";

	int = 1;

	int *=  8;		// int =  8
	int /= -3;		// int = -2
	printInt(); printString();

	float = 5.0;
```

```
	float *= 4;		// float =  20.0
	float /= int;	// float = -10.0
	printFloat();  printString();

	// can be powered only by number
	// A_vec4 * B_float = C_vec4
	// Ax * B = Cx
	// Ay * B = Cy
	// Az * B = Cz
	// Ak * B = Ck

	vec4(3.0);

	vec4 *= 2;
	printVec4(); // 3*2,3*2,3*2 = 6,6,6
	printString();

	vec4 /= 3;
	printVec4(); // 2,2,2
	printString();
}
```

Operators + and -

Works only with **numbers**(not bool), **vectors** and **strings**(only +)

+/- works the same way as +=/-= but it saves result in the class name variable

```
init{
	string = "\n";

	int = 1 + 3;
	printInt();		printString(); //4
	// int = 1 + 3 + 4; - doesn't work
	// maximum 2 numbers or vectors
	//
	// int += 1 + 2; - also doesn't work because it transforms into
	// int = 1 + 2;
	// int += int; - result will be 6
	// but
	// v_int = 4;
	// v_int += 3 + 2; - works. result: v_int = 9

	1.0 + 5.2; // "float =" is not necessary
	printFloat();	printString(); //6.2

	vec4(3.0);

	vec4 = vec4 + vec4;
	printVec4();	printString(); //6,6,6

	vec4 - vec4;
	printVec4();	printString(); //0,0,0
```

```
	"Cake4D" + "\n"; printString();
}
```

Operators * and /

Works only with **numbers**(not bool) ***not vectors**

*** vec4.x, vec3.y** and others count as numbers

& / works the same way as *= & /= but it saves result in the class name variable

```
init{
	string = "\n";

	int = 2 * 3;
	printInt();	printString(); //6

	1.0 / 2.5;
	printFloat();	printString(); //0.4

	vec4(3.0);

	vec4.x * 2;
	printFloat();	printString(); //6.0

	vec4.y / 1.5;
	printFloat();	printString(); //2.0

	vec4.x * vec4.y;
	printFloat();	printString(); //9.0
}
```

Operators <, >, <=, >=, ==, !=

(<, >, <=, >=) work with **numbers**(not bool)

(==, !=) work with **numbers**, **vectors** and **string**

```
	bool A;
	bool B;
init{
	string = "\n";
	A = true;
	B = false;

	A == B;
	printBool();	printString(); //false

	A != B;
	printBool();	printString(); //true

	"Cake4D" == "CAKE4D";
```

```
	printBool();	printString(); //false

	"4D" != "4D";
	printBool();	printString(); //false

	2 == 3;
	printBool();	printString(); //false

	2 != 3;
	printBool();	printString(); //true

	2 > 2;
	printBool();	printString(); //false

	2 < 3;
	printBool();	printString(); //true

	2 >= 2;
	printBool();	printString(); //true

	2 <= 2;
	printBool();	printString(); //true
}
```

Operators !, &&, ||

Works only with **bool**

```
	bool A;
	bool B;
init{
	string = "\n";
	A = true;
	B = false;

	A && B; // and
	printBool();	printString(); //false

	A || B; // or
	printBool();	printString(); //true

	!A;	// not
	printBool();	printString(); //false

	!B;	// not
	printBool();	printString(); //true
}
```

Operators |, &, ^, ~, >>, <<, >>>

Bitwise and Bit Shift Operators
Works only with **int**

```
int a;
int b;
int c;
init{
    a = 0xD;    //0000 1101 = 13
    b = 0x18;   //0001 1000 = 24
    a|b; // or
    printInt(); //0001 1101
    printString("\t= 29\n");
    a&b; //and
    printInt(); //0000 1000
    printString("\t= 8\n");
    a^b; //xor
    printInt(); //0001 0101
    printString("\t= 21\n");
    ~a;  //not
    printInt(); //1111 0010
    printString("\t= -14\n");
    c = 2;
    int >> c; //bit shift left
    printInt(); //1111 1100
    printString("\t= -4\n");
    a << c; //bit shift right
    printInt(); //0011 0100
    printString("\t= 52\n");
    int = -14;
    int >>> c; //shifts a zero into the leftmost position
    printInt(); //0011 1111  1111 1111   1111 1111  1111 1100
    printString("\t= 1073741820\n");
}
```

Arrays (@=, $=) and System Variables

```
/*
 *
 * Arrays:
 *
 * int arr_int[20];
 *
 * int = 5;         //  int[0] = 5;
 * arr_int @= int; //  int[$arr_int + int[0]] = int[0];
 * //  -> int[$arr_int + 5] = 5;
 *
 * int = 5;         //  int[0] = 5;
 * int $= arr_int; //  int[0] = int[$arr_int + int[0]];
 *
 *
 */

int  arr_int[6];
vec3 arr_vec3[6];

init{
      //array init

      int = 0; arr_int @= 5;
      int = 1; arr_int @= 4;
      int = 2; arr_int @= 3;
      int = 3; arr_int @= 2;
      int = 4; arr_int @= 1;
      int = 5; arr_int @= 0;

      int = 0; vec3(5.0); arr_vec3 @= vec3;
      int = 1; vec3(4.0); arr_vec3 @= vec3;
      int = 2; vec3(3.0); arr_vec3 @= vec3;
      int = 3; vec3(2.0); arr_vec3 @= vec3;
      int = 4; vec3(1.0); arr_vec3 @= vec3;
      int = 5; vec3(0.0); arr_vec3 @= vec3;

      //array using
      string = "\n";

      int = 0; int $= arr_int; printInt(); printString();
      int = 1; int $= arr_int; printInt(); printString();
      int = 2; int $= arr_int; printInt(); printString();
      int = 3; int $= arr_int; printInt(); printString();
      int = 4; int $= arr_int; printInt(); printString();
      int = 5; int $= arr_int; printInt(); printString();

      printString();
      int = 0; vec3 $= arr_vec3; printVec3(); printString();
      int = 1; vec3 $= arr_vec3; printVec3(); printString();
      int = 2; vec3 $= arr_vec3; printVec3(); printString();
      int = 3; vec3 $= arr_vec3; printVec3(); printString();
      int = 4; vec3 $= arr_vec3; printVec3(); printString();
      int = 5; vec3 $= arr_vec3; printVec3(); printString();
```

```
	//zero using
	// arr_int = arr_int[0]

	printString();
	int   = arr_int;	printInt();	printString();//5
	vec3 #= arr_vec3;	printVec3();	printString();//5.0, 5.0, 5.0
}

update{
	// time or delta is a time that has passed
	//   since the last update call

	//System variables normal access
	printFloat(time);    printString(); //time                  [s]
	printFloat(rtime);   printString(); //1.0/time              [1/s]
	printFloat(real_wh); printString(); //screen_w/screen_h     [-]
	printFloat(real_hw); printString(); //screen_h/screen_w     [-]

	//System variables array access
	int = 0; float $= float; printFloat(); printString(); //float
	int = 1; float $= float; printFloat(); printString(); //time
	int = 2; float $= float; printFloat(); printString(); //rtime
	int = 3; float $= float; printFloat(); printString(); //real_wh
	int = 4; float $= float; printFloat(); printString(); //real_hw

	quit();
}
```

You can access any **variable** or **link** using **$=** or **@=** operators. For this purpose you just need to use such class name system variables as **int**, **float**, **obj4d**, **vec4** and etc. as array name.

null can be used with **@=** operator to dispose the **object** link inside array

```
int = 5; arr_vec3 @= null; vec3 $= arr_vec3;
printVec3(); printString(" = null \n");
```

Cycles Examples

```
      float f;
      int i;
init{
      // bool - if it is true, than compiler changes
      // Current command line by its ID (int)
      if(bool , int);

      //Cycle: while(f < 2.0) { f += 1.0; }
      f = 0.0;

      LINK: link_while_again;
            f < 2.0; !bool; if (link_while_out);

                  f += 1.0;
                  printFloat(f); printString(" - while cycle\n");

            if (true, link_while_again);
      LINK : link_while_out;

      //Cycle: for(int i = 0; i<10; i++){}

      i = 0; LINK: link_for_again;
            i < 10; !bool; if (link_for_out);

                  printInt(i); printString(" - for cycle\n");

            i += 1; if (true, link_for_again);
      LINK : link_for_out;
}
```

System Functions

Print Functions

```
init{
	string = "Cake4D\n"; printString();

	bool = true; printBool(); printString(" - true \n");
	string  = bool; string += " - true  \n"; printString();

	int = 185;

	printInt();	printString(" - 185	\n");
	printIntHex();	printString(" - b9	\n");
	printIntOct();	printString(" - 271	\n");
	printIntBin();	printString(" - 10111001 \n");
	string  = int; string += " - 185	\n"; printString();

	vec2(2.0); printVec2(); printString(" - vec4 \n");
	vec3(3.0); printVec3(); printString(" - vec3 \n");
	vec4(4.0); printVec4(); printString(" - vec2 \n");

	string  = vec2; string += " - vec2 \n";
	string += vec3; string += " - vec3 \n";
	string += vec4; string += " - vec4 \n";
	printString();

	float = 0.3956;
	printFloat();  printString(" - 0.3956 \n");
	// int - number of digits after .
	formatFloat(2); string += " - 0.40 \n"; printString();

	string  = float; string += " - 0.3956 \n"; printString();
}
```

Math Functions

```
init{
	//Algebra
	//Returns the absolute value of a float value
	float = abs(-1.0);
	printFloat(); printString(" = 1.0\n");
	//Returns the square root of a float value
	float = sqrt(9.0);
	printFloat(); printString(" = 3.0\n");
	//Returns the natural logarithm e of a float
	float = log(1.0);
	printFloat(); printString(" = 0.0\n");
	//Returns the base 10 logarithm of a float value
	float = log10(100.0);
	printFloat(); printString(" = 2.0\n");
```

```
//The base of the natural logarithms
float = getE();
printFloat(); printString(" = 2.72\n");

//Trigonometry
//Returns the trigonometric sine of an angle
float = sin(1.0);
printFloat(); printString(" = 0.84\n");
//Returns the trigonometric cosine of an angle
float = cos(1.0);
printFloat(); printString(" = 0.54\n");
//The ratio of the circumference of a circle to its
float = getPI();
printFloat(); printString(" = 3.14\n");

//Random numbers
//A pseudorandom float greater than or equal
// to 0.0 and less than 1.0
float = random();
printFloat(); printString(" = [0.0, 1.0)\n");
//Returns a pseudorandom, uniformly distributed int value
// between 0 (inclusive) and the specified value (exclusive)
int = randomInt(5);
printInt(); printString(" = [0, 5)\n");

//Vectors math
//Returns the length of the vec4
vec4(1.0);
float = vec4_length();
printFloat(); printString(" = 2.0\n");
//Returns the length of the vec3
vec3(1.0);
float = vec3_length();
printFloat(); printString(" = 1.73\n");

//Rewrite a vec4 with the same direction but with length 1
vec4_norm();
printVec4(); printString(" = 0.50,0.50,0.50\n");
//Rewrite a vec3 with the same direction but with length 1
vec3_norm();
printVec3(); printString(" = 0.58,0.58,0.58\n");

vec3.x = 1.0; vec3.y = 0.0; vec3.z = -1.0;
vec4.x = 1.5; vec4.y = 0.5; vec4.z = -0.5; vec4.k = -1.5;
//Returns a maximum value among vec4 components
float = vec4_max();
printFloat(); printString(" = 1.5\n");
//Returns a minimum value among vec3 components
float = vec3_max();
printFloat(); printString(" = 1.0\n");
//Returns a maximum value among vec4 components
float = vec4_min();
printFloat(); printString(" = -1.5\n");
//Returns a minimum value among vec3 components
float = vec3_min();
printFloat(); printString(" = -1.0\n");
```

```
	//isCross();
	//Return true if point A with size dA touches B with size dB
	//bool = vec4.isCross();

	vec4.x = 1.0; // A center
	vec4.y = 1.0; // A size
	// A will be [0.5, 1.5]

	vec4.z = 2.0; // B center
	vec4.k = 1.0; // B size
	// B will be [1.5, 2.5]

	bool = isCross();
	printBool(); printString(" = A & B have crossed \n");

	vec4.x = 1.0; // A center
	vec4.y = 0.5; // A size
	// A will be [0.75, 1.25]

	vec4.z = 2.0; // B center
	vec4.k = 0.5; // B size
	// B will be [1.75, 2.25]

	bool = isCross();
	printBool(); printString(" = A & B have not crossed \n");
}
```

Graphic Card Functions

```
init{
	// Camera 4D
	cam4d();
	setCamera(); //draw_camera #= cam4d
	getCamera(); //cam4d #= draw_camera

	//FOV
	setFOV(80.0); //Sets field of view in angle degrees
	getFOV();	printFloat(); printString(" - FOV\n");

	// Sets camera maximum distance
	setFOV_FAR(300.0);
	getFOV_FAR();	printFloat(); printString(" - FOV_FAR\n");

	// Sets fog max distance (less than FOV_FAR)
	setFogDistance(200.0);
	// Sets sea fog max distance (less than FOV_FAR)
	setSeaFogDistance(20.0);

	// Sends FOV data to graphic card
	updateFOV();

	//FOG Color
	vec3(1.0);
```

```
        setFogColor(vec3);    // vec3 RGB
        setSeaFogColor1(vec3);      // vec3 Top    Color
        setSeaFogColor2(vec3);      // vec3 Bottom Color
        setSeaLevel1(0.0);    // sea start
        setSeaLevel2(1.0);    // sea end
        setSeaAmbient(true);  // Sets sea ambient effect
        // Sends FOG Color data to graphic card
        updateSeaFog();

        //2D Effect
        get2DEffect();// vec3 #= 2d_effect;
        vec3.x = 1.0; // brightness
        vec3.y = 1.0; // contrast
        vec3.z = 0.0; // 1 - saturation

        //Shade Effect
        // Sets shade area 0.0 - 100%, 1.5 - 50% , 3.0 - 0%
        setShade(3.0);
        vec4(0.0); vec4.k = 1.0;
        setShadeColor(vec4); //vec4 RGBA
        // Sends Shade Effect data to graphic card
        updateShade();

        //Fullscreen
        setFullscreen(false);
        // Returns fullscreen state on or off
        getFullscreen();
        printBool(); printString(" - fullscreen off\n");
}
```

Omega Functions

```
        //Omega Functions

        //Attaches a save by its ID and loads it
        // int - save id
        loadSave(int);

        // Reload the attached Save File
        restartMap();

        //Release the attached Save
        restartSaveManager();

        // Quit the program
        quit();    // omega quit

        // string - script file name
        // bool - need to save after loading
        // if no save file is attached, one will be created
        loadNewMap(string, bool);

        // Save File (attached one)
```

```
//Sets timer on/off of the Save
setTimeFreeze(bool);

//Returns timer current value
float = getPlayTime();
// Classes that can be saved and load by save(), load() functions
// int, float, bool, vec2, vec3, vec4, string
//Saves variable by its script name into the Save
save(string);
//Loads variable by its script name from the Save
load(string);
//Deletes all saved data of the attached Save
clearSave();

// Save Files Manager

// Changes unattached saves

//Deletes save file by ID
deleteSave(int);
//Returns amount of saves
getSaveManagerSize();
//Sets Save to read its metadata (doesn't load anything)
setSaveManagerID(int);

//Metadata functions
string = getSaveDate();
string = getSaveName();
int = getSaveLevelID();
int = getSaveManagerDifficulty();
int = getSaveManagerCrystals();
float = getSaveManagerCakes();

//Examples:

//PM.oml - load menu
restartSaveManager();
loadNewMap("maps/menu.map", false);

//PM.oml - restart
restartMap();

//PM.oml - quit
quit();

//menu.map - init
clearSave();

//menu.map - easy mode start
game_difficulty = 0; save("game_difficulty");
loadNewMap("maps/level_h0.map", true);

//menu.map - normal mode start
game_difficulty = 1; save("game_difficulty");
loadNewMap("maps/level_p1.map", true);
```

```
	//menu.map - hard mode start
	game_difficulty = 2; save("game_difficulty");
	loadNewMap("maps/level_p1.map", true);

	//menu.map - load save
	loadSave(ui_load_saves_id);
```

Draw List Functions

```
	obj4d obj_a;
	obj4d obj_b;
init{
	//Draw list keeps links to the objects that were constructed
	// by class method.

	//Examples:
	obj4d(); //adds 1 item to draw list
	obj_a #= obj4d; // does nothing
	obj_b #= obj4d; // does nothing

	//Sky4D list
	//sky4d #= sky_darw_list[int];
	getSky4dById(int);

	//Obj4D list
	//obj4d #= obj_darw_list[int];
	getObj4dById(int);

	//Part4D list
	//part4d #= part_darw_list[int];
	getPart4dById(int);

	//Aobj4D list
	//aobj4d #= aobj_darw_list[int];
	getAobj4dById(int);
	//pers4d #= aobj_darw_list[int];
	getPers4dById(int);
}
```

Resources Functions

```
init{
	//Loads UI Texture
	loadTex2d("ui_texture.png");

	//setting sky texture parameters
	// size x size x size - 1 cube (sky has 8 of them)
	// texture should have 2D size as (size*size) x (size*8)
	// sky_64.png (4096x512) has size 64
	setSkyTexture(32, "sky_32.png");

	//Loads 3D texture for obj4d, aobj4d, pers4d, part4d
	// size x size x size - 1 cube (can have any number of cubes)
	setTexture(32, "texture_32.png");
```

```
	loadText("launcher.json");
	//Returns string from text resource by ID
	string = getStringByID(0);
	string += "\n"; printString();
	//Returns language code
	string = getLang();
	string += "\n"; printString();
}
```

Input Functions

```
init{
	//Returns true if the key[int] is pressed
	// int - key id
	// Use KEY.oml to get key id by name
	bool = getKey(int);
	// Returns mouse and scrolls speeds
	// x, y - mouse  x, y
	// z, k - scroll x, y
	vec4 = getMouse();

	//Returns true if the game window has focus
	bool = getFocus();

	//Additional

	//Bindings functions were fully implemented by PM.oml and menu.map.

	// Linker
	//action name - [0-3] - axis, [4-37] - buttons
	//to display key and axis settings
	//full name
	string = getLinkerActionName(int);
	//only first  binding
	string = getLinkerFirstActionName(int);
	//only second binding
	string = getLinkerSecondActionName(int);
	//to set new key or axis
	initLinker(int);
	//to be informed about possible key or axis choice conflict
	// States:
	// 0:   Ready
	// 1,2: Reading  binding   (1 - key, 2 - axis)
	// 3,4: Checking binding   (3 - key, 4 - axis)
	// 5,6: Conflict appeared  (5 - key, 6 - axis)
	// 7,8: Linking new binding(7 - key, 8 - axis)
	int = getLinkerState();
	//Returns id of conflict key or axis, to display it to the user
	int = getLinkerConflictActionID();
	//to solve key or axis choice conflict
	// true - add bindings
	// false - cancel bindings
	releaseLinker(bool);
```

```
	//Returns the sensitivity of axis
	// int = 00_0 - first two - axis [0-4], last is biding [0-1]
	// id: 00_0 -> 0 - mouse_x  - first  binding
	// id: 00_1 -> 1 - mouse_x  - second binding
	// id: 01_0 -> 2 - mouse_y  - first  binding
	// id: 11_0 -> 6 - scroll_y - first  binding
	float = getLinkerSensitivity(int);
	//Sets the sensitivity of axis
	setLinkerSensitivity(int, float);

	//Resets the bindings to default values
	resetLinker();
}
```

UI Functions

```
init{
	//Compass
	//int&0b0_11_00 - marginY
	//int&0b0_00_11 - marginX
	// 0 - left, 2 - right, 1 or 3 - center
	int = 1 + 4; //center
	// x - x position,
	// y - y position,
	// z - compass size (1.0 = h of display)
	vec3(0.0); vec3.z = 0.15;
	// bool - isVisible?
	setCompassData(int, vec3, bool);

	//Mouse
	//Visibility of the mouse
	mouseSetVisible(bool);
	//Set loading icon
	mouseSetLoading(bool);

	//Pers4D Monitoring
	//Sets pers4d to monitor it (press F2 to see monitoring)
	setPersonConsole(pers4d);

	//Indicator
	//Sets visibility of indicators
	setIndiVisible(bool);
	//Creates a new indicator
	//int - indicator ID
	int = createIndi();
	//Select indicator bi ID to edit it
	setIndi(int);
	//Sets the size of selected indicator
	setIndiSize(float);
	//Sets the visible distance of selected indicator
	setIndiDist(float);
	//Returns the link of the position of selected indicator
	//vec4 #= selected_indicator.position;
	getIndiPosition();
```

```
	//Returns the links of the indicator texture data:
	//vec4 #= selected_indicator.tex_offset_XY_size_ZK;
	//vec2 #= selected_indicator.tex_mult_XY;
	getIndiTexData();
	//Deletes the selected indicator
	deleteIndi();


	//Buttons
	//Creates a new button
	//in  int - string id from text file
	//out int - button id
	int = createButton(int);
	//Tries to select button by ID
	//in  int - button ID
	//out int - selected button ID
	int = setButton(int);
	//Sets visibility of selected button
	setButtonVisible(bool);
	//Sets button text id
	//int - string id from text file
	setButtonTextID(int);
	//Returns the link of the button position and size
	//vec4 #= selected_button.position_XY_size_ZK
	getButtonPointSize();
	//Sets the dxwh of selected button
	//dxwh - x distance that changes with screen height
	//x += dxwh*screen_w/screen_h
	setButtonDXHW(float);
	//Sets the texture data of selected button
	// vec4 - tex_offset_XY_size_ZK
	// vec2 - tex_mult_XY
	// int  - margin
	// int&0b0_11_00 - marginY
	// int&0b0_00_11 - marginX
	// 0 - left, 2 - right, 1 or 3 - center
	setButtonTexData(vec4, vec2, int);
	//Returns button affine transformation
	// vec4 #= selected_button.affine;
	// x' = a.x*x + a.y*y;
	// y' = a.z*x + a.k*y;
	getButtonAffine();

}
```

OpenAL Functions

OpenAL functions were implemented inside **ALF.oml**.

Use **ALF.oml** to set sounds and music.

```
init{
	//Returns the link to the audio gain
	//vec4 #= audio_gain;
```

```
        //x - system gain
        //y - effect gain
        //z - music  gain
        //k - voice  gain
        getAudioGain();
        vec4 #= null;

        //Use ALF.oml to set audio settings

        //OpenAL
        //Runs Audio function
        //int - function id
        alRunFunc(int);
        //Selects device by id
        alSetDevice(int);
        //Selects source by id
        // bool - true if source was selected
        bool = alSetSource(int);
        //Selects buffer by id
        alSetBuffer(int);
        //Returns buffer id by its name
        int = alGetBuffer(string);
}
```