TOUHOU U.F.O. (UNDEFINED FANTASTIC OBJECT)

Proyecto Final b) **Juego de Consola**

TEORIA DE LA COMPUTACION

MARIA MAGDALENA MURILLO LEAÑO

Requisitos Minimos del Sistema

Windows: XP/Vista/7/8/10

ESPACIO LIBRE EN DISCO DURO: 200MB

MEMORIA RAM: 2GB

PROCESADOR: INTEL PENTIUM

TARJETA DE VIDEO: 256MB

東方星蓮船
Undefined Fantastic Object
12th Touhou Project
Shooting Game for Windows XP/2000
Chase Flying Fantastic Object !

Integrantes

Encinas Mardueño Christopher Brad

Peña Solis José Pablo

Cruz Torres Gustavo Eduardo

FECHA DE ENTREGA 13/05/2019

OBJETIVO

El objetivo del desarrollo de esta aplicación es hacer uso de los autómatas y ver su funcionamiento dentro ya sea de un videojuego o sistema real. Los autómatas que pueden ser utilizados son:

Máquina de Estado Finito (MEF) o **Autómata de Estado Finito** (AEF) o **Autómata Pila** (AP).

INTRODUCCION

Decidimos realizar un fanmade game de una de las sagas mas populares dentro de los Bullet Hell llamada Touhou, un shooter vertical. El motivo fue que es un juego que no requiere muchos recursos, puede ser jugado con facilidad, y tiene un gran atractivo visual debido a los patrones de los ataques y su gama de colores, pero a su vez con una dificultad muy elevada.

SOLUCION DE LA APLICACIÓN

Utilizamos un engine basado en C que se ejecuta mediante scripts llamado “Danmakufu” el cual nos permitió crear un shooter vertical. El automata utilizado para representar el funcionamiento del juego fue una Maquina de Estado Finito (MEF), el cual nos muestra las diferentes transiciones utilizadas en el juego. Una de las ventajas de utilizar este engine es que es altamente personalizable por lo que al ser un juego de código libre cualquier jugador puede personalizarlo a su gusto con los conocimientos básicos, además los requisitos minimos para poder utilizarlo son minimos por lo que prácticamente cualquier computadora puede ejecutarlo y hacer uso de el. Los patrones de los ataques fueron realizados mediante angulos y ecuaciones paramétricas para patrones mas complejos.

MAQUINA DE ESTADO FINITO

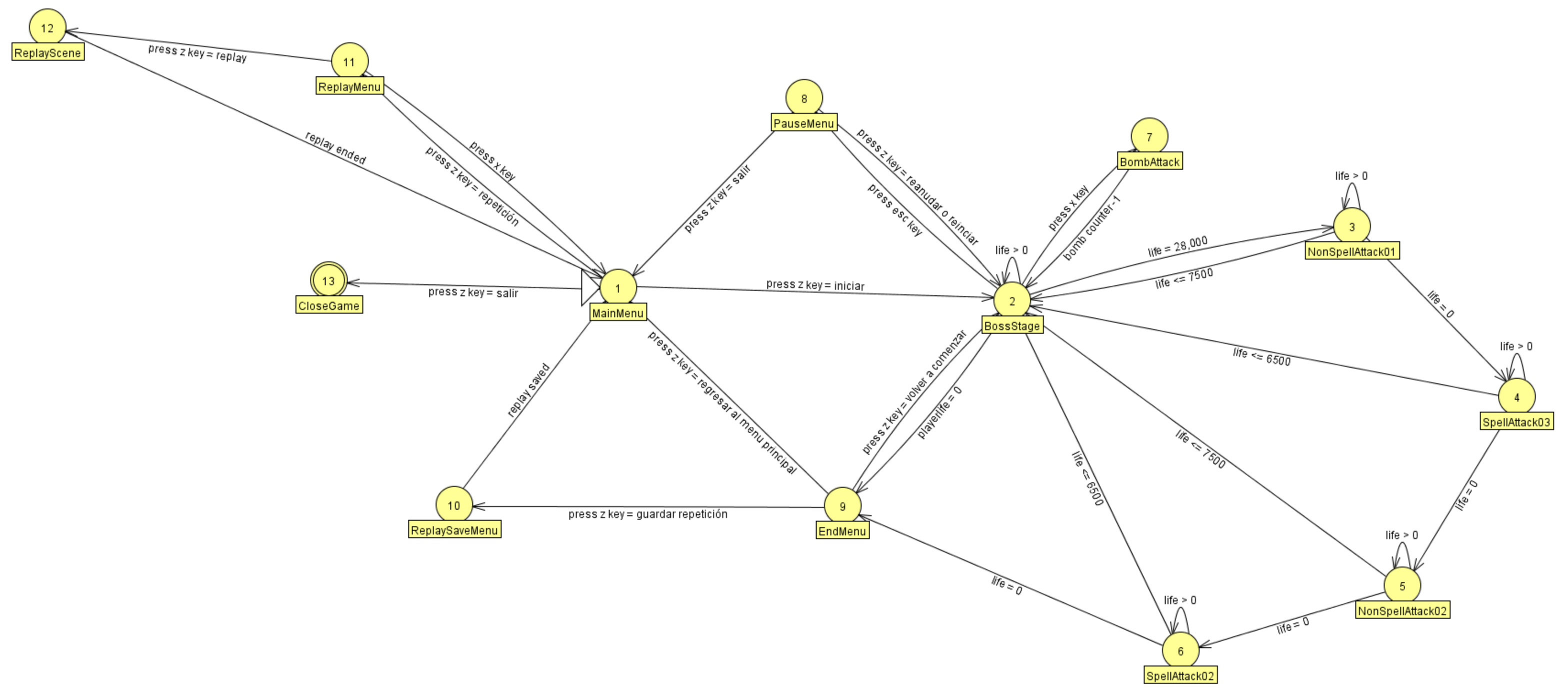


TABLA DE TRANSICION DE ESTADOS

Tabla de Transiciones												
Estados	Entradas											
	Z key	X key	Esc key	Life = 28,00	Life > 0	Life = 0	Playerlife = 0	Replay saved	Replay ended	Bomb counter -1	Life <= 7500	Life <= 6500
1	2, 11, 13	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ
2	λ	7	8	3	2	λ	9	λ	λ	λ	λ	λ
3	λ	λ	λ	λ	3	4	λ	λ	λ	λ	2	λ
4	λ	λ	λ	λ	4	5	λ	λ	λ	λ	λ	2
5	λ	λ	λ	λ	5	6	λ	λ	λ	λ	2	λ
6	λ	λ	λ	λ	6	9	λ	λ	λ	λ	λ	2
7	λ	λ	λ	λ	λ	λ	λ	λ	λ	2	λ	λ
8	1, 2	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ
9	1, 2, 10	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ
10	λ	λ	λ	λ	λ	λ	λ	1	λ	λ	λ	λ
11	12	1	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ
12	λ	λ	λ	λ	λ	λ	λ	λ	1	λ	λ	λ
13	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ

CODIGO FUENTE

Package_Main.txt

```
#TouhouDanmakufu[Package]
#Title["Undefined Fantastic Object"]
#Text["Byakuren Boss"]
#Player["./player/The Ultimate Player (Set 1)/A Ultimate.txt"]
#System["./system/System.txt"]

InstallFont(GetCurrentScriptDirectory() ~ "font/PintoLunaire.ttf");
InstallFont(GetCurrentScriptDirectory() ~ "font/Oswald-Regular.ttf");
InstallFont(GetCurrentScriptDirectory() ~ "font/AsakuraSlab.ttf");

let music = ObjSound_Create();
ObjSound_Load(music, GetCurrentScriptDirectory() ~ "./se/Main Menu Theme
- A Shadow in the Blue Sky.ogg");
    ObjSound_SetSoundDivision(music, SOUND_BGM);
    ObjSound_SetVolumeRate(music, 100);
    ObjSound_SetLoopEnable(music, true);
    ObjSound_SetLoopTime(music, 24, 240.31);

let music2 = ObjSound_Create();
    ObjSound_Load(music2, GetCurrentScriptDirectory() ~ "./se/Heian
Alien.ogg");
    ObjSound_SetSoundDivision(music2, SOUND_BGM);
    ObjSound_SetVolumeRate(music2, 100);
    ObjSound_SetLoopEnable(music2, false);
    ObjSound_SetLoopTime(music2, 24, 240.31);

@Initialize
{
        TTitleScene();
}

@MainLoop
{
        yield;
}

task TTitleScene
{
        ObjSound_Stop(music2);
        ObjSound_Play(music);
        let bEndTitleScene = false;
        let dir = GetCurrentScriptDirectory();
        let pathTitle = dir ~ "./img/TitleMenu.png";

        let objTitleImage = ObjPrim_Create(OBJ_SPRITE_2D);
        Obj_SetRenderPriorityI(objTitleImage, 20);
        ObjPrim_SetTexture(objTitleImage, pathTitle);
        ObjSprite2D_SetSourceRect(objTitleImage, 0, 0, 640, 479);
        ObjSprite2D_SetDestRect(objTitleImage, 0, 0, 640, 479);

        let objTitleText = ObjText_Create();
        ObjText_SetFontType(objTitleText, "PintoLunaire");
        ObjText_SetText(objTitleText, "Touhou Project 12");
        ObjText_SetFontSize(objTitleText, 30);
        ObjText_SetFontBold(objTitleText, true);
        ObjText_SetFontColorTop(objTitleText, 255, 102, 102);
        ObjText_SetFontColorBottom(objTitleText, 255, 255, 255);
        ObjText_SetFontBorderType(objTitleText, BORDER_FULL);
        ObjText_SetFontBorderColor(objTitleText, 51, 0, 0);
        ObjText_SetFontBorderWidth(objTitleText, 2);
        Obj_SetRenderPriorityI(objTitleText, 30);
        ObjRender_SetX(objTitleText, 355); //342
        ObjRender_SetY(objTitleText, 390); //375

        let objTitleText2 = ObjText_Create();
        ObjText_SetFontType(objTitleText2, "AsakuraSlab");
        ObjText_SetText(objTitleText2, "Undefined Fantastic Object");
        ObjText_SetFontSize(objTitleText2, 34);
        ObjText_SetFontBold(objTitleText2, true);
        ObjText_SetFontColorTop(objTitleText2, 51, 153, 255);
        ObjText_SetFontColorBottom(objTitleText2, 255, 255, 255); // 204 0
102
        ObjText_SetFontBorderType(objTitleText2, BORDER_FULL);
        ObjText_SetFontBorderColor(objTitleText2, 0, 51, 102);
        ObjText_SetFontBorderWidth(objTitleText2, 2);
        Obj_SetRenderPriorityI(objTitleText2, 30);
        ObjRender_SetX(objTitleText2, 168);
        ObjRender_SetY(objTitleText2, 425); //350

        let objTitleText3 = ObjText_Create();
        ObjText_SetFontType(objTitleText3, "AsakuraSlab");
        ObjText_SetText(objTitleText3, "Fanmade Game  v1.0");
        ObjText_SetFontSize(objTitleText3, 13);
        ObjText_SetFontBold(objTitleText3, true);
        ObjText_SetFontColorTop(objTitleText3, 0, 0, 0);
        ObjText_SetFontColorBottom(objTitleText3, 0, 0, 0);
```

```
Obj_SetRenderPriorityI(objTitleText3, 30);
ObjRender_SetX(objTitleText3, 5);
ObjRender_SetY(objTitleText3, 465);

let objTitleText4 = ObjText_Create();
ObjText_SetFontType(objTitleText4, "AsakuraSlab");
ObjText_SetText(objTitleText4, "1");
ObjText_SetFontSize(objTitleText4, 25);
ObjText_SetFontBold(objTitleText4, true);
ObjText_SetFontColorTop(objTitleText4, 255, 102, 102);
ObjText_SetFontColorBottom(objTitleText4, 255, 255, 255);
ObjText_SetFontBorderType(objTitleText4, BORDER_FULL);
ObjText_SetFontBorderColor(objTitleText4, 51, 0, 0);
ObjText_SetFontBorderWidth(objTitleText4, 2);
Obj_SetRenderPriorityI(objTitleText4, 30);
ObjRender_SetX(objTitleText4, 610);
ObjRender_SetY(objTitleText4, 10);

let INDEX_START = 0;
let INDEX_REPLAY = 1;
let INDEX_QUIT = 2;
let INDEX_TITLE = 3;
let selectIndex = 3;
task TMenuItem(let index, let mx, let my, let text)
{
        function CreateTextObject(let mx, let my, let text)
        {
                let obj = ObjText_Create();
                ObjText_SetFontType(obj, "AsakuraSlab");
                ObjText_SetText(obj, text);
                ObjText_SetFontSize(obj, 30);
                ObjText_SetFontBold(obj, true);
                ObjText_SetFontColorTop(obj, 0, 153, 0);
                ObjText_SetFontColorBottom(obj, 0, 153, 0);

                Obj_SetRenderPriorityI(obj, 30);
                ObjRender_SetX(obj, mx);
                ObjRender_SetY(obj, my);
                return obj;
        }

        let objText = CreateTextObject(mx, my, text);
        let objSelect = CreateTextObject(mx, my, text);
        ObjRender_SetBlendType(objSelect, BLEND_ADD_RGB);
        while(!bEndTitleScene)
        {
                Obj_SetVisible(objSelect, index == selectIndex);
                yield;
        }
        Obj_Delete(objText);
        Obj_Delete(objSelect);
}

let mx = 450;
let my = 116;
let texts = ["Iniciar", "Repetición", "Salir", " "];
var countMenu = length(texts);
ascent(var iText in 0 .. countMenu)
{
        TMenuItem(iText, mx, my, texts[iText]);
        my += 32;
}

while(GetVirtualKeyState(VK_OK) != KEY_FREE){yield;}

let frameKeyHold = 0;
loop
{
        if(GetVirtualKeyState(VK_OK) == KEY_PUSH)
        {
                if(selectIndex == INDEX_TITLE)
                {
                        TTitleScene();
                }
                if(selectIndex == INDEX_START)
                {
                        TStageScene("");
                }
                else if(selectIndex == INDEX_REPLAY)
                {
                        TReplaySelectScene();
                }
                else if(selectIndex == INDEX_QUIT)
                {
                        ClosePackage();
                }
                break;
        }

        if(GetVirtualKeyState(VK_UP) == KEY_PUSH)
        {
                selectIndex--;
                }
                else if(GetVirtualKeyState(VK_DOWN) == KEY_PUSH)
                {
                        selectIndex++;
                }
                else if(GetVirtualKeyState(VK_UP) == KEY_HOLD)
                {
                        frameKeyHold++;
                        if(frameKeyHold == 30 || (frameKeyHold > 30 &&
(frameKeyHold % 10 == 0)))
                        {
                                selectIndex--;
                        }
                }
                else if(GetVirtualKeyState(VK_DOWN) == KEY_HOLD)
                {
                        frameKeyHold++;
                        if(frameKeyHold == 30 || (frameKeyHold > 30 &&
(frameKeyHold % 10 == 0)))
                        {
                                selectIndex++;
                        }
                }
                else
                {
                        frameKeyHold = 0;
                }

                if(selectIndex < 0)
                {
                        selectIndex = countMenu - 1;
                }
                else
                {
                        selectIndex %= countMenu;
                }

                yield;
        }

        bEndTitleScene = true;
        Obj_Delete(objTitleImage);
        Obj_Delete(objTitleText);
        Obj_Delete(objTitleText2);
        Obj_Delete(objTitleText3);
        Obj_Delete(objTitleText4);
```

```
}

task TStageScene(let pathReplay)
{
        ObjSound_Stop(music);
        ObjSound_Play(music2);
        let dirCurrent = GetCurrentScriptDirectory();
        let dirModule = GetModuleDirectory();
        let pathMainScript = dirCurrent ~ "main.dnh";
        let pathPlayer = dirCurrent ~ "./player/The Ultimate Player (Set
1)/A Ultimate.txt";
        RenderSceneToTransitionTexture();
        TTransition();
        InitializeStageScene();
        if(length(pathReplay) > 0)
        {
                SetStageReplayFile(pathReplay);
        }

        let indexStage = 1;
        SetStageIndex(indexStage);
        SetStageMainScript(pathMainScript);
        SetStagePlayerScript(pathPlayer);
        StartStageScene();
        loop
        {
                if(GetVirtualKeyState(VK_PAUSE) == KEY_PUSH)
                {
                        let resPause = RunPauseScene();
                        alternative(resPause)
                        case(RESULT_RETRY)
                        {
                                if(!IsReplay())
                                {
                                        TerminateStageScene();
                                        TStageScene("");
                                        return;
                                }
                        }
                        case(RESULT_END)
                        {
                                TerminateStageScene();
                        }
                }

                if(!IsReplay() && GetKeyState(KEY_BACK) == KEY_PUSH)
```

```
{

        TerminateStageScene();
        TStageScene("");
        return;
}

let stgSceneState = GetStageSceneState();
if(stgSceneState == STAGE_STATE_FINISHED)
{

        let stageResult = GetStageSceneResult();
        alternative(stageResult)
        case(STAGE_RESULT_CLEARED)
        {

                if(indexStage == 1)
                {

                        TEndScene();
                        break;
                }
                else
                {

                        indexStage++;
                        SetStageIndex(indexStage);
                        SetStageMainScript(pathMainScript);
                        StartStageScene();
                        TTransition();
                }
        }
        case(STAGE_RESULT_PLAYER_DOWN)
        {

                TEndScene();
                break;
        }
        case(STAGE_RESULT_BREAK_OFF)
        {


                TTitleScene();
                break;
        }
}
```

```
                yield;
        }

        TTransition();
}

task TEndScene()
{
        if(IsReplay())
        {
                TTitleScene();
                return;
        }

        FinalizeStageScene();

        let dirModule = GetCurrentScriptDirectory();
        let pathScript = dirModule ~ "system/EndScene.txt";
        let idScript = LoadScript(pathScript);
        StartScript(idScript);

        while(!IsCloseScript(idScript))
        {
                yield;
        }

        let result = GetScriptResult(idScript);
        alternative(result)
        case(RESULT_SAVE_REPLAY)
        {

                TReplaySaveScene();
        }
        case(RESULT_END)
        {

                TTitleScene();
        }
        case(RESULT_RETRY)
        {

                TStageScene("");
        }
}

function RunPauseScene()
```

```
{
        RenderSceneToTransitionTexture();
        PauseStageScene(true);

        let dirModule = GetCurrentScriptDirectory();
        let pathScript = dirModule ~ "system/Pause.txt";

        let idScript = LoadScript(pathScript);
        StartScript(idScript);

        while(!IsCloseScript(idScript))
        {
                yield;
        }

        PauseStageScene(false);

        let res = GetScriptResult(idScript);
        return res;
}

task TReplaySelectScene()
{
        let objTitleText5 = ObjText_Create();
        ObjText_SetFontType(objTitleText5, "AsakuraSlab");
        ObjText_SetText(objTitleText5, "11");
        ObjText_SetFontSize(objTitleText5, 25);
        ObjText_SetFontBold(objTitleText5, true);
        ObjText_SetFontColorTop(objTitleText5, 255, 102, 102);
        ObjText_SetFontColorBottom(objTitleText5, 255, 255, 255);
        ObjText_SetFontBorderType(objTitleText5, BORDER_FULL);
        ObjText_SetFontBorderColor(objTitleText5, 51, 0, 0);
        ObjText_SetFontBorderWidth(objTitleText5, 2);
        Obj_SetRenderPriorityI(objTitleText5, 30);
        ObjRender_SetX(objTitleText5, 600);
        ObjRender_SetY(objTitleText5, 10);

        let dirCurrent = GetCurrentScriptDirectory();
        let pathScript = dirCurrent ~ "Package_ReplaySelectScene.txt";

        let idScript = LoadScript(pathScript);
        StartScript(idScript);

        while(!IsCloseScript(idScript))
        {
                yield;
        }

        let result = GetScriptResult(idScript);
        if(length(result) == 0)
        {
                TTitleScene();
        }
        else
        {
                TStageScene(result);
        }
        Obj_Delete(objTitleText5);
}

task TReplaySaveScene()
{
        let dirModule = GetCurrentScriptDirectory();
        let pathScript = dirModule ~ "system/ReplaySaveScene.txt";

        let idScript = LoadScript(pathScript);
        StartScript(idScript);

        while(!IsCloseScript(idScript))
        {
                yield;
        }

        TTitleScene();
}

function RenderSceneToTransitionTexture()
{

        let textureName = GetTransitionRenderTargetName();
        RenderToTextureA1(textureName, 0, 100, true);
}

task TTransition
{
        let textureName = GetTransitionRenderTargetName();

        let objImage = ObjPrim_Create(OBJ_SPRITE_2D);
        Obj_SetRenderPriorityI(objImage, 100);
        ObjPrim_SetTexture(objImage, textureName);
        ObjSprite2D_SetSourceRect(objImage, 0, 0, 640, 480);
        ObjSprite2D_SetDestRect(objImage, 0, 0, 640, 480);
        let alpha = 255;
        while(alpha > 0)
        {
                ObjRender_SetAlpha(objImage, alpha);
                alpha -= 16;
                yield;
        }
        Obj_Delete(objImage);
}
```

Main.dnh

```
#TouhouDanmakufu[Plural]
#ScriptVersion[3]
#System["./system/System.txt"]
#Player["./player/The Ultimate Player (Set 1)/A Ultimate.txt"]
#Title["Cosmical Mind"]
#Text["Byakuren Boss"]
#Background["./background.txt"]

@Event
{}

@Finalize
{}

@Initialize
{
    TPlural;
}

@MainLoop
{
    yield;
}

task TPlural
{
        let dir = GetCurrentScriptDirectory();
    let obj = ObjEnemyBossScene_Create();
    ObjEnemyBossScene_Add(obj, 0, dir ~ "./NonSpellAttack01.dnh");
    ObjEnemyBossScene_Add(obj, 0, dir ~ "./SpellAttack03.dnh");
    ObjEnemyBossScene_Add(obj, 1, dir ~ "./NonSpellAttack02.dnh");
    ObjEnemyBossScene_Add(obj, 1, dir ~ "./SpellAttack02.dnh");
    ObjEnemyBossScene_LoadInThread(obj);
    ObjEnemyBossScene_Regist(obj);
    while(!Obj_IsDeleted(obj)){
        yield;
    }
    CloseScript(GetOwnScriptID());
}
```

NonSpellAttack01.dnh

```
#System["./system/System.txt"]
#include "./system/ShotConst.txt"
#include "./GizmoSpriteLibrary.txt"
#include "./KyunBullet_Const.txt"
#include "./selibrary.txt"

let objBoss;
let objScene = GetEnemyBossSceneObjectID();

@Event
{
        alternative(GetEventType())
        case(EV_REQUEST_LIFE){
                SetScriptResult(7500)
        }
        case(EV_REQUEST_TIMER){
                SetScriptResult(60)
        }
}

@Initialize
{
        objBoss = ObjEnemy_Create(OBJ_ENEMY_BOSS);
        ObjEnemy_Regist(objBoss);
        ObjMove_SetDestAtFrame(objBoss, GetCenterX(), 60, 60);

        renderNueUFO(objBoss);
        Obj_SetValue(objBoss, "cast", 0);
        TFinalize;
        MainTask;
}

@MainLoop
{
        ObjEnemy_SetIntersectionCircleToShot(objBoss,
ObjMove_GetX(objBoss), ObjMove_GetY(objBoss), 32);
```

```
        ObjEnemy_SetIntersectionCircleToPlayer(objBoss,
ObjMove_GetX(objBoss), ObjMove_GetY(objBoss), 24);
        yield;
}

task MainTask
{
        let objTitleText = ObjText_Create();
        ObjText_SetFontType(objTitleText, "AsakuraSlab");
        ObjText_SetText(objTitleText, "2");
        ObjText_SetFontSize(objTitleText, 25);
        ObjText_SetFontBold(objTitleText, true);
        ObjText_SetFontColorTop(objTitleText, 255, 102, 102);
        ObjText_SetFontColorBottom(objTitleText, 255, 255, 255);
        ObjText_SetFontBorderType(objTitleText, BORDER_FULL);
        ObjText_SetFontBorderColor(objTitleText, 51, 0, 0);
        ObjText_SetFontBorderWidth(objTitleText, 2);
        Obj_SetRenderPriorityI(objTitleText, 30);
        ObjRender_SetX(objTitleText, 360);
        ObjRender_SetY(objTitleText, 415);
        wait(120);
        Movement;
        DamageSound;
        Obj_Delete(objTitleText);
        Fire;
        BulletRing
}

task DamageSound{
        while(ObjEnemy_GetInfo(objBoss, INFO_LIFE) > 0){
        if(ObjEnemy_GetInfo(objBoss, INFO_SHOT_HIT_COUNT) >
0){SE_Play(dam2, 60);}
        wait(1);
}
}

task Fire
{
        let objTitleText = ObjText_Create();
        ObjText_SetFontType(objTitleText, "AsakuraSlab");
        ObjText_SetText(objTitleText, "3");
        ObjText_SetFontSize(objTitleText, 25);
        ObjText_SetFontBold(objTitleText, true);
        ObjText_SetFontColorTop(objTitleText, 255, 102, 102);
        ObjText_SetFontColorBottom(objTitleText, 255, 255, 255);
        ObjText_SetFontBorderType(objTitleText, BORDER_FULL);
        ObjText_SetFontBorderColor(objTitleText, 51, 0, 0);
        ObjText_SetFontBorderWidth(objTitleText, 2);
        Obj_SetRenderPriorityI(objTitleText, 30);
        ObjRender_SetX(objTitleText, 360);
        ObjRender_SetY(objTitleText, 415);

        while(ObjEnemy_GetInfo(objBoss, INFO_LIFE) > 0){
                let angleT = rand(0, 360);
                loop(30){
                        ascent(i in 0..2){
                let ObjBullet = CreateShotA1(ObjMove_GetX(objBoss),
ObjMove_GetY(objBoss), 3, angleT, 1627, 5);
                ObjMove_AddPatternA1(ObjBullet, 60, 2-5, 360*cos(angleT));
        }
                angleT += 360/30;
        }
                wait(120);
        }
}

task BulletRing
{
        while(ObjEnemy_GetInfo(objBoss, INFO_LIFE) > 0){
        let angleT = 0;
        wait(60);
    loop(30){
        CreateShotA1(ObjMove_GetX(objBoss) + 140*cos(angleT),
ObjMove_GetY(objBoss) + 140*sin(angleT), 2-5, angleT, 1614, 5);
        angleT += 360/30;
    }
    wait(60);
    loop(35){
        CreateShotA1(ObjMove_GetX(objBoss) + 90*cos(angleT),
ObjMove_GetY(objBoss) + 90*sin(angleT), 2-5, angleT, 1614, 5);
        angleT += 360/35;
    }
    wait(60);
    loop(30){
        CreateShotA1(ObjMove_GetX(objBoss) + 45*cos(angleT),
ObjMove_GetY(objBoss) + 45*sin(angleT), 2-5, angleT, 1614, 5);
        angleT += 360/30;
    }
    wait(60);
        }
}
```

```
task Movement
{
        while(ObjEnemy_GetInfo(objBoss, INFO_LIFE) > 0){
                ObjMove_SetDestAtFrame(objBoss, rand(GetCenterX() + 90,
GetCenterX() - 90), rand(GetCenterY() -60, GetCenterY() - 120), 60);
                        wait(120);
        }
}

task TFinalize
{
        while(ObjEnemy_GetInfo(objBoss, INFO_LIFE) > 0) {yield;}
                if(ObjEnemyBossScene_GetInfo(objScene,
INFO_PLAYER_SHOOTDOWN_COUNT)
                        +ObjEnemyBossScene_GetInfo(objScene,
INFO_PLAYER_SHOOTDOWN_COUNT) == 0){
                        AddScore(ObjEnemyBossScene_GetInfo(objScene,
INFO_SPELL_SCORE));
                }
                Obj_Delete(objBoss);
                DeleteShotAll(TYPE_ALL, TYPE_IMMEDIATE);
                SetAutoDeleteObject(true);
                CloseScript(GetOwnScriptID());
}

function GetCenterX()
{
        return GetStgFrameWidth() / 2;
}

function GetCenterY()
{
        return GetStgFrameHeight() / 2;
}

function rand_int(min, max)
{
        return round(rand(min, max))
}

function wait(n){
        loop(n){yield;}
}
```

SpellAttack03.dnh

```
#System["./system/System.txt"]
#include "./system/ShotConst.txt"
#include "./GizmoSpriteLibrary.txt"
#include "./KyunBullet_Const.txt"
#include "./selibrary.txt"
#include "./Cutin.txt"

let objBoss;
let objScene = GetEnemyBossSceneObjectID();

@Event
{
        alternative(GetEventType())
        case(EV_REQUEST_LIFE){
                SetScriptResult(6500)
        }
        case(EV_REQUEST_TIMER){
                SetScriptResult(60)
        }
        case(EV_REQUEST_SPELL_SCORE){
                SetScriptResult(10000000)
        }
}

@Initialize
{
        objBoss = ObjEnemy_Create(OBJ_ENEMY_BOSS);
        ObjEnemy_Regist(objBoss);
        ObjMove_SetDestAtFrame(objBoss, GetCenterX(), 60, 60);

        ObjEnemyBossScene_StartSpell(objScene);

        StartSpell;
        renderNueUFO(objBoss);
        Obj_SetValue(objBoss, "cast", 1);
        TFinalize;
        MainTask;
}

@MainLoop
{
        ObjEnemy_SetIntersectionCircleToShot(objBoss,
ObjMove_GetX(objBoss), ObjMove_GetY(objBoss), 32);
```

```
        ObjEnemy_SetIntersectionCircleToPlayer(objBoss,
ObjMove_GetX(objBoss), ObjMove_GetY(objBoss), 24);
        yield;
}

task MainTask
{
        DamageSound;
        wait(120);
        BorderOfLife;
}

task DamageSound{
        while(ObjEnemy_GetInfo(objBoss, INFO_LIFE) > 0){
        if(ObjEnemy_GetInfo(objBoss, INFO_SHOT_HIT_COUNT) >
0){SE_Play(dam2, 60);}
        wait(1);
}
}

task TFinalize
{
        while(ObjEnemy_GetInfo(objBoss, INFO_LIFE) > 0) {yield;}
                if(ObjEnemyBossScene_GetInfo(objScene,
INFO_PLAYER_SHOOTDOWN_COUNT)
                        +ObjEnemyBossScene_GetInfo(objScene,
INFO_PLAYER_SHOOTDOWN_COUNT) == 0){
                        AddScore(ObjEnemyBossScene_GetInfo(objScene,
INFO_SPELL_SCORE));
                }
                Obj_Delete(objBoss);
                DeleteShotAll(TYPE_ALL, TYPE_IMMEDIATE);
                SetAutoDeleteObject(true);
                CloseScript(GetOwnScriptID());
}

task BorderOfLife
{
        let objTitleText = ObjText_Create();
        ObjText_SetFontType(objTitleText, "AsakuraSlab");
        ObjText_SetText(objTitleText, "4");
        ObjText_SetFontSize(objTitleText, 25);
        ObjText_SetFontBold(objTitleText, true);
        ObjText_SetFontColorTop(objTitleText, 255, 102, 102);
        ObjText_SetFontColorBottom(objTitleText, 255, 255, 255);
        ObjText_SetFontBorderType(objTitleText, BORDER_FULL);
        ObjText_SetFontBorderColor(objTitleText, 51, 0, 0);
        ObjText_SetFontBorderWidth(objTitleText, 2);
        Obj_SetRenderPriorityI(objTitleText, 30);
        ObjRender_SetX(objTitleText, 360);
        ObjRender_SetY(objTitleText, 415);

        let angleT = rand(0, 360);
        let objCount = 0;
        loop{
                SE_Play(shot1, 40);
                loop(5){
                        let obj = CreateShotA1(ObjMove_GetX(objBoss),
ObjMove_GetY(objBoss), 3, angleT, 1, 10);
                        angleT += 360/5;
                }
                angleT += sin(objCount) * 6;
                objCount++;
                yield;
        }
}

task StartSpell
{
        let objBoss = GetEnemyBossObjectID[0];
        loop(90){yield;}

        let Nue = GetCurrentScriptDirectory() ~ "./img/NueSpellcard.png";
        ObjCutin_SetSpellcardS4("Surprising Rain \"Guerrilla Typhoon\"",
Nue, BYAKUREN, 255, 104, 104);
        ObjCutin_LaunchS3(BYAKUREN, Nue, "Normal");

        ObjEnemyBossScene_StartSpell(GetEnemyBossObjectID);
}

function GetCenterX()
{
        return GetStgFrameWidth() / 2;
}

function GetCenterY()
{
        return GetStgFrameHeight() / 2;
}

function rand_int(min, max)
{
        return round(rand(min, max))
}

function wait(n){
        loop(n){yield;}
}
```

NonSpellAttack02.dnh

```
#System["./system/System.txt"]
#include "./system/ShotConst.txt"
#include "./GizmoSpriteLibrary.txt"
#include "./KyunBullet_Const.txt"
#include "./selibrary.txt"

let objBoss;
let objScene = GetEnemyBossSceneObjectID();

@Event
{
        alternative(GetEventType())
        case(EV_REQUEST_LIFE){
                SetScriptResult(7500)
        }
        case(EV_REQUEST_TIMER){
                SetScriptResult(60)
        }
}

@Initialize
{
        objBoss = ObjEnemy_Create(OBJ_ENEMY_BOSS);
        ObjEnemy_Regist(objBoss);
        ObjMove_SetDestAtFrame(objBoss, GetCenterX(), 60, 60);

        renderNueUFO(objBoss);
        Obj_SetValue(objBoss, "cast", 0);
        TFinalize;
        MainTask;
}

@MainLoop
{
        ObjEnemy_SetIntersectionCircleToShot(objBoss,
ObjMove_GetX(objBoss), ObjMove_GetY(objBoss), 32);
```

```
        ObjEnemy_SetIntersectionCircleToPlayer(objBoss,
ObjMove_GetX(objBoss), ObjMove_GetY(objBoss), 24);
        yield;
}

task MainTask
{
        wait(120);
        Movement;
        DamageSound;
        ParametricFire;
}

task DamageSound{
        while(ObjEnemy_GetInfo(objBoss, INFO_LIFE) > 0){
        if(ObjEnemy_GetInfo(objBoss, INFO_SHOT_HIT_COUNT) >
0){SE_Play(dam2, 60);}
        wait(1);
}
}

task ParametricFire
{
        let objTitleText = ObjText_Create();
        ObjText_SetFontType(objTitleText, "AsakuraSlab");
        ObjText_SetText(objTitleText, "5");
        ObjText_SetFontSize(objTitleText, 25);
        ObjText_SetFontBold(objTitleText, true);
        ObjText_SetFontColorTop(objTitleText, 255, 102, 102);
        ObjText_SetFontColorBottom(objTitleText, 255, 255, 255);
        ObjText_SetFontBorderType(objTitleText, BORDER_FULL);
        ObjText_SetFontBorderColor(objTitleText, 51, 0, 0);
        ObjText_SetFontBorderWidth(objTitleText, 2);
        Obj_SetRenderPriorityI(objTitleText, 30);
        ObjRender_SetX(objTitleText, 360);
        ObjRender_SetY(objTitleText, 415);

        while(ObjEnemy_GetInfo(objBoss, INFO_LIFE) > 0){
        let angleT = rand(0, 360);
        wait(120);
        loop(60){
                loop(3){
                        let obj = CreateShotA2(ObjMove_GetX(objBoss) +
120*cos(angleT * 1) - cos(angleT * 7) ^ 3, ObjMove_GetY(objBoss) +
90*sin(angleT * 1) - sin(angleT * 7) ^ 3, 0, angleT * 3, 3, 2, 1625, 10);
                        angleT += 360/3;
                }
                angleT += 5;
                yield;
        }
}
}

task Movement
{
        while(ObjEnemy_GetInfo(objBoss, INFO_LIFE) > 0){
                ObjMove_SetDestAtFrame(objBoss, 60, 120, 60);
                wait(360);
                ObjMove_SetDestAtFrame(objBoss, GetCenterX(), 120, 60);
                wait(360);
                ObjMove_SetDestAtFrame(objBoss, 320, 120, 60);
                wait(360);
                ObjMove_SetDestAtFrame(objBoss, GetCenterX(), 120, 60);
                wait(360);
        }
}


task TFinalize
{
        while(ObjEnemy_GetInfo(objBoss, INFO_LIFE) > 0) {yield;}
                if(ObjEnemyBossScene_GetInfo(objScene,
INFO_PLAYER_SHOOTDOWN_COUNT)
                        +ObjEnemyBossScene_GetInfo(objScene,
INFO_PLAYER_SHOOTDOWN_COUNT) == 0){
                        AddScore(ObjEnemyBossScene_GetInfo(objScene,
INFO_SPELL_SCORE));
                }
                Obj_Delete(objBoss);
                DeleteShotAll(TYPE_ALL, TYPE_IMMEDIATE);
                SetAutoDeleteObject(true);
                CloseScript(GetOwnScriptID());
}

function GetCenterX()
{
        return GetStgFrameWidth() / 2;
}

function GetCenterY()
{
        return GetStgFrameHeight() / 2;
```

```
}

function rand_int(min, max)
{
        return round(rand(min, max))
}

function wait(n){
        loop(n){yield;}
}
```

SpellAttack02.dnh

```
#System["./system/System.txt"]
#include "./system/ShotConst.txt"
#include "./GizmoSpriteLibrary.txt"
#include "./KyunBullet_Const.txt"
#include "./selibrary.txt"
#include "./Cutin.txt"

let objBoss;
let objScene = GetEnemyBossSceneObjectID();

@Event
{
        alternative(GetEventType())
        case(EV_REQUEST_LIFE){
                SetScriptResult(6500)
        }
        case(EV_REQUEST_TIMER){
                SetScriptResult(60)
        }
        case(EV_REQUEST_SPELL_SCORE){
                SetScriptResult(10000000)
        }
}

@Initialize
{
        objBoss = ObjEnemy_Create(OBJ_ENEMY_BOSS);
        ObjEnemy_Regist(objBoss);
        ObjMove_SetDestAtFrame(objBoss, GetCenterX(), 60, 60);

        ObjEnemyBossScene_StartSpell(objScene);

        StartSpell;
        renderNueUFO(objBoss);
        Obj_SetValue(objBoss, "cast", 1);
        TFinalize;
        MainTask;
}

@MainLoop
{
        ObjEnemy_SetIntersectionCircleToShot(objBoss,
ObjMove_GetX(objBoss), ObjMove_GetY(objBoss), 32);
        ObjEnemy_SetIntersectionCircleToPlayer(objBoss,
ObjMove_GetX(objBoss), ObjMove_GetY(objBoss), 24);
        yield;
}

task MainTask
{
        DamageSound;
        PurpleMist;
}

task DamageSound{
        while(ObjEnemy_GetInfo(objBoss, INFO_LIFE) > 0){
        if(ObjEnemy_GetInfo(objBoss, INFO_SHOT_HIT_COUNT) >
0){SE_Play(dam2, 60);}
        wait(1);
}
}

task TFinalize
{
        while(ObjEnemy_GetInfo(objBoss, INFO_LIFE) > 0) {yield;}
                if(ObjEnemyBossScene_GetInfo(objScene,
INFO_PLAYER_SHOOTDOWN_COUNT)
                        +ObjEnemyBossScene_GetInfo(objScene,
INFO_PLAYER_SHOOTDOWN_COUNT) == 0){
                        AddScore(ObjEnemyBossScene_GetInfo(objScene,
INFO_SPELL_SCORE));
                }
                Obj_Delete(objBoss);
                DeleteShotAll(TYPE_ALL, TYPE_IMMEDIATE);
                SetAutoDeleteObject(true);
                CloseScript(GetOwnScriptID());
}

task PurpleMist
```

```
{
        let objTitleText = ObjText_Create();
        ObjText_SetFontType(objTitleText, "AsakuraSlab");
        ObjText_SetText(objTitleText, "6");
        ObjText_SetFontSize(objTitleText, 25);
        ObjText_SetFontBold(objTitleText, true);
        ObjText_SetFontColorTop(objTitleText, 255, 102, 102);
        ObjText_SetFontColorBottom(objTitleText, 255, 255, 255);
        ObjText_SetFontBorderType(objTitleText, BORDER_FULL);
        ObjText_SetFontBorderColor(objTitleText, 51, 0, 0);
        ObjText_SetFontBorderWidth(objTitleText, 2);
        Obj_SetRenderPriorityI(objTitleText, 30);
        ObjRender_SetX(objTitleText, 360);
        ObjRender_SetY(objTitleText, 415);

        ObjMove_SetDestAtFrame(objBoss, GetCenterX(), 112, 60);
        wait(120);
        let angleT = rand(0, 360);
        let objCount = 0;
        loop{
                SE_Play(shot1, 40);
                loop(6){
                        ascent(i in 0..3){
                        if(ObjEnemy_GetInfo(objBoss, INFO_LIFE) <=
0){return;}
                        let obj = CreateShotA1(ObjMove_GetX(objBoss),
ObjMove_GetY(objBoss), 2, angleT, 47, 10);
                        angleT += 360/6;
                }
        }
                angleT += 10*sin(objCount) * 20*cos(objCount);
                objCount++;
                yield;
        }
}

task StartSpell
{
        let objBoss = GetEnemyBossObjectID[0];
        loop(90){yield;}

        let Nue = GetCurrentScriptDirectory() ~ "./img/NueSpellcard.png";
        ObjCutin_SetSpellcardS4("Ominous Clouds \"Heian Dark Clouds\"",
Nue, BYAKUREN, 255, 104, 104);
        ObjCutin_LaunchS3(BYAKUREN, Nue, "Normal");
        ObjEnemyBossScene_StartSpell(GetEnemyBossObjectID);
}

function GetCenterX()
{
        return GetStgFrameWidth() / 2;
}

function GetCenterY()
{
        return GetStgFrameHeight() / 2;
}

function rand_int(min, max)
{
        return round(rand(min, max))
}

function wait(n){
        loop(n){yield;}
}
```

Effect.txt

```
task TExplosionA(x, y, dAlpha, dScale)
{
        let path = GetCurrentScriptDirectory() ~ "system/img/Effect.png";
        let obj = ObjPrim_Create(OBJ_SPRITE_2D);
        ObjPrim_SetTexture(obj, path);
        Obj_SetRenderPriority(obj, 0.65);
        ObjRender_SetBlendType(obj, BLEND_ADD_RGB);
        ObjSprite2D_SetSourceRect(obj, 1, 1, 63, 63);
        ObjSprite2D_SetDestCenter(obj);
        ObjRender_SetPosition(obj, x, y, 0);

        let scale = 0;
        let alpha = 255;
        while(alpha > 0)
        {
                ObjRender_SetColor(obj, alpha, alpha, alpha);
                ObjRender_SetScaleXYZ(obj, scale, scale, 1);

                scale += dScale;
                alpha -= dAlpha;
                yield;
        }
        Obj_Delete(obj);
}
```

EndScene.txt

```
@Initialize
{
        SetAutoDeleteObject(true);
        TBackground();
        TMenu();
}

@MainLoop
{
        yield;
}

@Finalize
{}

task TBackground
{
        task TVertex(var index, var left, var top, var right, var bottom)
        {
                ObjPrim_SetVertexPosition(obj, index + 0, left, top, 0);
                ObjPrim_SetVertexPosition(obj, index + 1, left, bottom,
0);
                ObjPrim_SetVertexPosition(obj, index + 2, right, top, 0);
                ObjPrim_SetVertexPosition(obj, index + 3, right, top, 0);
                ObjPrim_SetVertexPosition(obj, index + 4, left, bottom,
0);
                ObjPrim_SetVertexPosition(obj, index + 5, right, bottom,
0);

                ObjPrim_SetVertexUVT(obj, index + 0, left, top);
                ObjPrim_SetVertexUVT(obj, index + 1, left, bottom);
                ObjPrim_SetVertexUVT(obj, index + 2, right, top);
                ObjPrim_SetVertexUVT(obj, index + 3, right, top);
                ObjPrim_SetVertexUVT(obj, index + 4, left, bottom);
                ObjPrim_SetVertexUVT(obj, index + 5, right, bottom);

                if(left >= 32 && right <= 416 && top >=16 && bottom <=
464)
                {
                        let alpha = 255;
                        while(alpha >= 128)
                        {
                                ObjPrim_SetVertexAlpha(obj, index + 0,
alpha);
                                ObjPrim_SetVertexAlpha(obj, index + 1,
alpha/2);
                                ObjPrim_SetVertexAlpha(obj, index + 2,
alpha/2);
                                ObjPrim_SetVertexAlpha(obj, index + 3,
alpha/2);
                                ObjPrim_SetVertexAlpha(obj, index + 4,
alpha/2);
                                ObjPrim_SetVertexAlpha(obj, index + 5,
alpha);
                                alpha -= 255 / frame;

                                yield;
                        }
                }
        }

        let frame = 30;
        let countH = 20;
        let countV = 30;
        let width = 640 / countH;
        let height = 480 / countV;
        let target = GetTransitionRenderTargetName();
        let obj = ObjPrim_Create(OBJ_PRIMITIVE_2D);
        ObjPrim_SetPrimitiveType(obj, PRIMITIVE_TRIANGLELIST);
        ObjPrim_SetVertexCount(obj, countH * countV * 6);
        Obj_SetRenderPriorityI(obj, 0);
        ObjPrim_SetTexture(obj, target);

        ascent(ix in 0.. countH)
        {
                ascent(iy in 0.. countV)
                {
                        let index = (ix + iy * countH) * 6;
                        let left = ix * width;
                        let right = left + width;
                        let top = iy * height;
                        let bottom = top + height;
                        TVertex(index, left, top, right, bottom);
                }
        }
}

task TMenu
```

```
{
        let selectIndex = 0;
        task TMenuItem(let index, let mx, let my, let text)
        {
                function CreateTextObject(let mx, let my, let text)
                {
                        let obj = ObjText_Create();
                        ObjText_SetFontType(obj, "AsakuraSlab");
                        ObjText_SetText(obj, text);
                        ObjText_SetFontSize(obj, 23);
                        ObjText_SetFontBold(obj, true);
                        ObjText_SetFontColorTop(obj, 160, 160, 160);
                        ObjText_SetFontColorBottom(obj, 255, 255, 255);
                        ObjText_SetFontBorderType(obj, BORDER_FULL);
                        ObjText_SetFontBorderColor(obj,0, 0, 0);
                        ObjText_SetFontBorderWidth(obj, 2);
                        Obj_SetRenderPriorityI(obj, 10);
                        ObjRender_SetX(obj, mx);
                        ObjRender_SetY(obj, my);
                        return obj;
                }

                let objText = CreateTextObject(mx, my, text);
                let objSelect = CreateTextObject(mx, my, text);
                ObjRender_SetBlendType(objSelect, BLEND_ADD_RGB);
                loop
                {
                        Obj_SetVisible(objSelect, index == selectIndex);
                        yield;
                }
        }

        let objTitleText = ObjText_Create();
        ObjText_SetFontType(objTitleText, "AsakuraSlab");
        ObjText_SetText(objTitleText, "9");
        ObjText_SetFontSize(objTitleText, 25);
        ObjText_SetFontBold(objTitleText, true);
        ObjText_SetFontColorTop(objTitleText, 255, 102, 102);
        ObjText_SetFontColorBottom(objTitleText, 255, 255, 255);
        ObjText_SetFontBorderType(objTitleText, BORDER_FULL);
        ObjText_SetFontBorderColor(objTitleText, 51, 0, 0);
        ObjText_SetFontBorderWidth(objTitleText, 2);
        Obj_SetRenderPriorityI(objTitleText, 30);
        ObjRender_SetX(objTitleText, 392);
        ObjRender_SetY(objTitleText, 430);
        let mx = 48;
        let my = 32;
        let texts = ["Guardar Repetición", "Regresar al Menu Principal",
"Volver a Comenzar"];
        var countMenu = length(texts);
        ascent(var iText in 0 .. countMenu)
        {
                TMenuItem(iText, mx, my, texts[iText]);
                my += 32;
        }

        while(GetVirtualKeyState(VK_OK) != KEY_FREE){yield;}

        let frameKeyHold = 0;
        loop
        {
                if(GetVirtualKeyState(VK_OK) == KEY_PULL)
                {
                        let listResult = [RESULT_SAVE_REPLAY, RESULT_END,
RESULT_RETRY];
                        SetScriptResult(listResult[selectIndex]);
                        CloseScript(GetOwnScriptID());
                        return;
                }

                if(GetVirtualKeyState(VK_UP) == KEY_PUSH)
                {
                        selectIndex--;
                }
                else if(GetVirtualKeyState(VK_DOWN) == KEY_PUSH)
                {
                        selectIndex++;
                }
                else if(GetVirtualKeyState(VK_UP) == KEY_HOLD)
                {
                        frameKeyHold++;
                        if(frameKeyHold == 30 || (frameKeyHold > 30 &&
(frameKeyHold % 10 == 0)))
                        {
                                selectIndex--;
                        }
                }
                else if(GetVirtualKeyState(VK_DOWN) == KEY_HOLD)
                {
                        frameKeyHold++;
```

```
			if(frameKeyHold == 30 || (frameKeyHold > 30 &&
(frameKeyHold % 10 == 0)))
			{
				selectIndex++;
			}
		}
		else
		{
			frameKeyHold = 0;
		}

		if(selectIndex < 0)
		{
			selectIndex = countMenu - 1;
		}
		else
		{
			selectIndex %= countMenu;
		}

		yield;
	}
}
```

Pause.txt

```
@Initialize
{
	SetAutoDeleteObject(true);
	TBackground();
	TMenu();
}

@MainLoop
{
	yield;
}

@Finalize
{}

task TBackground
{
	task TVertex(var index, var left, var top, var right, var bottom)
	{
		ObjPrim_SetVertexPosition(obj, index + 0, left, top, 0);
```

```
                ObjPrim_SetVertexPosition(obj, index + 1, left, bottom, 
0);
                ObjPrim_SetVertexPosition(obj, index + 2, right, top, 0);
                ObjPrim_SetVertexPosition(obj, index + 3, right, top, 0);
                ObjPrim_SetVertexPosition(obj, index + 4, left, bottom, 
0);
                ObjPrim_SetVertexPosition(obj, index + 5, right, bottom, 
0);

                ObjPrim_SetVertexUVT(obj, index + 0, left, top);
                ObjPrim_SetVertexUVT(obj, index + 1, left, bottom);
                ObjPrim_SetVertexUVT(obj, index + 2, right, top);
                ObjPrim_SetVertexUVT(obj, index + 3, right, top);
                ObjPrim_SetVertexUVT(obj, index + 4, left, bottom);
                ObjPrim_SetVertexUVT(obj, index + 5, right, bottom);

                if(left >= 32 && right <= 416 && top >=16 && bottom <= 
464)
                {
                        let alpha = 255;
                        while(alpha >= 128)
                        {
                                ObjPrim_SetVertexAlpha(obj, index + 0, 
alpha);
                                ObjPrim_SetVertexAlpha(obj, index + 1, 
alpha/2);
                                ObjPrim_SetVertexAlpha(obj, index + 2, 
alpha/2);
                                ObjPrim_SetVertexAlpha(obj, index + 3, 
alpha/2);
                                ObjPrim_SetVertexAlpha(obj, index + 4, 
alpha/2);
                                ObjPrim_SetVertexAlpha(obj, index + 5, 
alpha);
                                alpha -= 255 / frame;

                                yield;
                        }
                }
        }

        let frame = 30;
        let countH = 20;
        let countV = 30;
        let width = 640 / countH;
        let height = 480 / countV;
```

```
		let target = GetTransitionRenderTargetName();
		let obj = ObjPrim_Create(OBJ_PRIMITIVE_2D);
		ObjPrim_SetPrimitiveType(obj, PRIMITIVE_TRIANGLELIST);
		ObjPrim_SetVertexCount(obj, countH * countV * 6);
		Obj_SetRenderPriorityI(obj, 0);
		ObjPrim_SetTexture(obj, target);

		ascent(ix in 0.. countH)
		{
			ascent(iy in 0.. countV)
			{
				let index = (ix + iy * countH) * 6;
				let left = ix * width;
				let right = left + width;
				let top = iy * height;
				let bottom = top + height;
				TVertex(index, left, top, right, bottom);
			}
		}

	}

	task TMenu
	{
		let selectIndex = 0;
		task TMenuItem(let index, let mx, let my, let text)
		{
			function CreateTextObject(let mx, let my, let text)
			{
				let obj = ObjText_Create();
				ObjText_SetFontType(obj, "AsakuraSlab");
				ObjText_SetText(obj, text);
				ObjText_SetFontSize(obj, 23);
				ObjText_SetFontBold(obj, true);
				ObjText_SetFontColorTop(obj, 160, 160, 160);
				ObjText_SetFontColorBottom(obj, 255, 255, 255);
				ObjText_SetFontBorderType(obj, BORDER_FULL);
				ObjText_SetFontBorderColor(obj,0, 0, 0);
				ObjText_SetFontBorderWidth(obj, 2);
				Obj_SetRenderPriorityI(obj, 10);
				ObjRender_SetX(obj, mx);
				ObjRender_SetY(obj, my);
				return obj;
			}

			let objText = CreateTextObject(mx, my, text);
			let objSelect = CreateTextObject(mx, my, text);
			ObjRender_SetBlendType(objSelect, BLEND_ADD_RGB);
			loop
			{
				Obj_SetVisible(objSelect, index == selectIndex);
				yield;
			}
		}

		let objTitleText = ObjText_Create();
		ObjText_SetFontType(objTitleText, "AsakuraSlab");
		ObjText_SetText(objTitleText, "8");
		ObjText_SetFontSize(objTitleText, 25);
		ObjText_SetFontBold(objTitleText, true);
		ObjText_SetFontColorTop(objTitleText, 255, 102, 102);
		ObjText_SetFontColorBottom(objTitleText, 255, 255, 255);
		ObjText_SetFontBorderType(objTitleText, BORDER_FULL);
		ObjText_SetFontBorderColor(objTitleText, 51, 0, 0);
		ObjText_SetFontBorderWidth(objTitleText, 2);
		Obj_SetRenderPriorityI(objTitleText, 30);
		ObjRender_SetX(objTitleText, 264);
		ObjRender_SetY(objTitleText, 415);

		let mx = 48;
		let my = 32;
		let texts = ["REANUDAR", "SALIR", "REINICIAR"];
		var countMenu = length(texts);
		ascent(var iText in 0 .. countMenu)
		{
			TMenuItem(iText, mx, my, texts[iText]);
			my += 32;
		}

		while(GetVirtualKeyState(VK_PAUSE) != KEY_FREE){yield;}

		let frameKeyHold = 0;
		loop
		{
			if(GetVirtualKeyState(VK_OK) == KEY_PULL)
			{
				let listResult = [RESULT_CANCEL, RESULT_END,
	RESULT_RETRY];
				SetScriptResult(listResult[selectIndex]);
				CloseScript(GetOwnScriptID());
				return;
			}
		if(GetVirtualKeyState(VK_CANCEL) == KEY_PULL ||
GetVirtualKeyState(VK_PAUSE) == KEY_PULL)
		{
			SetScriptResult(RESULT_CANCEL);
			CloseScript(GetOwnScriptID());
			return;
		}

		if(GetVirtualKeyState(VK_UP) == KEY_PUSH)
		{
			selectIndex--;
		}
		else if(GetVirtualKeyState(VK_DOWN) == KEY_PUSH)
		{
			selectIndex++;
		}
		else if(GetVirtualKeyState(VK_UP) == KEY_HOLD)
		{
			frameKeyHold++;
			if(frameKeyHold == 30 || (frameKeyHold > 30 &&
(frameKeyHold % 10 == 0)))
			{
				selectIndex--;
			}
		}
		else if(GetVirtualKeyState(VK_DOWN) == KEY_HOLD)
		{
			frameKeyHold++;
			if(frameKeyHold == 30 || (frameKeyHold > 30 &&
(frameKeyHold % 10 == 0)))
			{
				selectIndex++;
			}
		}
		else
		{
			frameKeyHold = 0;
		}

		if(selectIndex < 0)
		{
			selectIndex = countMenu - 1;
		}
		else
		{
```

```
                        selectIndex %= countMenu;
                }

                yield;
        }
}
```

ReplaySaveScene.txt

```
@Initialize
{
        SetAutoDeleteObject(true);
        LoadReplayList();

        TBackground();
        TReplayIndexSelection();
}

@MainLoop
{
        yield;
}

@Finalize
{
}

let MENU_INDEX_SELECTION = 1;
let MENU_NAME_ENTRY = 2;
let menuMode = MENU_INDEX_SELECTION;

function CreateTextObject(let mx, let my, let size, let text)
{
        let obj = ObjText_Create();
        ObjText_SetText(obj, text);
        ObjText_SetFontSize(obj, size);
        ObjText_SetFontBold(obj, true);
        ObjText_SetFontColorTop(obj, 128, 128, 255);
        ObjText_SetFontColorBottom(obj, 64, 64, 255);
        ObjText_SetFontBorderType(obj, BORDER_FULL);
        ObjText_SetFontBorderColor(obj,255, 255, 255);
        ObjText_SetFontBorderWidth(obj, 2);
        Obj_SetRenderPriorityI(obj, 10);
        ObjRender_SetX(obj, mx);
        ObjRender_SetY(obj, my);
        return obj;
}
```

```
task TBackground
{
        let target = GetTransitionRenderTargetName();
        let obj = ObjPrim_Create(OBJ_SPRITE_2D);
        ObjPrim_SetTexture(obj, target);
        Obj_SetRenderPriority(obj, 0.1);
        ObjSprite2D_SetSourceRect(obj, 0, 0, 640, 480);
        ObjSprite2D_SetDestCenter(obj);
        ObjRender_SetPosition(obj, 320, 240, 0);
        ObjRender_SetAlpha(obj, 64);
}

task TReplayIndexSelection()
{
        let cursorY = 0;
        let page = 0;
        let countMaxItem = REPLAY_INDEX_DIGIT_MAX - REPLAY_INDEX_DIGIT_MIN
+ 1;
        let countItemPerPage = 10;
        let pageMax = trunc((countMaxItem - 1) / countItemPerPage);
        pageMax = max(pageMax, 1);
        let lastPageMaxCursorY = trunc(countMaxItem / countItemPerPage);

        task TMenuItem(let itemY)
        {
                let objTitleText = ObjText_Create();
                ObjText_SetFontType(objTitleText, "AsakuraSlab");
                ObjText_SetText(objTitleText, "10");
                ObjText_SetFontSize(objTitleText, 25);
                ObjText_SetFontBold(objTitleText, true);
                ObjText_SetFontColorTop(objTitleText, 255, 102, 102);
                ObjText_SetFontColorBottom(objTitleText, 255, 255, 255);
                ObjText_SetFontBorderType(objTitleText, BORDER_FULL);
                ObjText_SetFontBorderColor(objTitleText, 51, 0, 0);
                ObjText_SetFontBorderWidth(objTitleText, 2);
                Obj_SetRenderPriorityI(objTitleText, 30);
                ObjRender_SetX(objTitleText, 385);
                ObjRender_SetY(objTitleText, 430);

                let objText = CreateTextObject(64, 64 + 30 * itemY, 18,
"");
                let objSelect = CreateTextObject(64, 64 + 30 * itemY, 18,
"");
                ObjRender_SetBlendType(objSelect, BLEND_ADD_RGB);
                let oldPage = -1;
                while(menuMode == MENU_INDEX_SELECTION)
                {
                        if(page != oldPage)
                        {
                                let index = page * countItemPerPage + itemY
        + 1;
                                let text = rtos("00", index) ~ " ";
                                if(IsValidReplayIndex(index))
                                {
                                        text = text ~ vtos("-8s",
        GetReplayInfo(index, REPLAY_USER_NAME)) ~ " ";
                                        text = text ~ GetReplayInfo(index,
        REPLAY_DATE_TIME) ~ " ";
                                        text = text ~ rtos("000000000000",
        GetReplayInfo(index, REPLAY_TOTAL_SCORE)) ~ " ";
                                }
                                else
                                {
                                        text = text ~ "No Data";
                                }
                                ObjText_SetText(objText, text);
                                ObjText_SetText(objSelect, text);
                                oldPage = page;
                        }

                        if(page == pageMax && itemY >= lastPageMaxCursorY)
                        {
                                Obj_SetVisible(objText, false);
                                Obj_SetVisible(objSelect, false);
                        }
                        else
                        {
                                Obj_SetVisible(objText, true);
                                Obj_SetVisible(objSelect, itemY ==
        cursorY);
                        }

                        yield;
                }
                Obj_Delete(objText);
                Obj_Delete(objSelect);
                Obj_Delete(objTitleText);
        }

        ascent(let iItem in 0 .. countItemPerPage)
        {
                TMenuItem(iItem);
        }

        while(GetVirtualKeyState(VK_OK) != KEY_FREE){yield;}

        let frameKeyHold = 0;
        while(menuMode == MENU_INDEX_SELECTION)
        {
                if(GetVirtualKeyState(VK_OK) == KEY_PULL)
                {
                        menuMode = MENU_NAME_ENTRY;
                        let index = page * countItemPerPage + cursorY + 1;
                        TNameEntry(index);

                        break;
                }

                if(GetVirtualKeyState(VK_UP) == KEY_PUSH ||
GetVirtualKeyState(VK_UP) == KEY_HOLD)
                {
                        frameKeyHold++;
                        if(GetVirtualKeyState(VK_UP) == KEY_PUSH ||
                                frameKeyHold == 20 ||
                                 (frameKeyHold > 20 && (frameKeyHold % 10
== 0)))
                        {
                                cursorY--;
                        }
                }
                else if(GetVirtualKeyState(VK_DOWN) == KEY_PUSH ||
GetVirtualKeyState(VK_DOWN) == KEY_HOLD)
                {
                        frameKeyHold++;
                        if(GetVirtualKeyState(VK_DOWN) == KEY_PUSH ||
                                frameKeyHold == 20 ||
                                 (frameKeyHold > 20 && (frameKeyHold % 10
== 0)))
                        {
                                cursorY++;
                        }
                }
                else if(GetVirtualKeyState(VK_LEFT) == KEY_PUSH ||
GetVirtualKeyState(VK_LEFT) == KEY_HOLD)
                {
                        frameKeyHold++;
                        if(GetVirtualKeyState(VK_LEFT) == KEY_PUSH ||
                                frameKeyHold == 20 ||
                                 (frameKeyHold > 20 && (frameKeyHold % 10
== 0)))
                        {
                                page--;
                        }
                }
                else if(GetVirtualKeyState(VK_RIGHT) == KEY_PUSH ||
GetVirtualKeyState(VK_RIGHT) == KEY_HOLD)
                {
                        frameKeyHold++;
                        if(GetVirtualKeyState(VK_RIGHT) == KEY_PUSH ||
                                frameKeyHold == 20 ||
                                 (frameKeyHold > 20 && (frameKeyHold % 10
== 0)))
                        {
                                page++;
                        }
                }
                else
                {
                        frameKeyHold = 0;
                }

                if(page < 0)
                {
                        page = pageMax;
                }
                else if(page > pageMax)
                {
                        page = 0;
                }

                if(page != pageMax)
                {
                        if(cursorY < 0)
                        {
                                cursorY = countItemPerPage - 1;
                        }
                        else if(cursorY >= countItemPerPage)
                        {
                                cursorY = 0;
                        }
                }
                else
                {
                        if(cursorY < 0)
                        {
                                cursorY = lastPageMaxCursorY - 1;
                        }
                        else if(cursorY >= lastPageMaxCursorY)
                        {
                                cursorY = 0;
                        }
                }

                yield;
        }

}


task TNameEntry(let replayIndex)
{
        let strTextIn =
        [

        ["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P"],

        ["Q","R","S","T","U","V","W","X","Y","Z",".",",",":",";","_","@"],

        ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p"],
                ["q","r","s","t","u","v","w","x","y","z","+","-
","/","*","=","%"],

        ["0","1","2","3","4","5","6","7","8","9","0","!","?","'","\"","$"]
,
                ["（","）","｛","｝","［","］","<",">","&","#","｜","~","^","　","
","Fin"]
        ];

        let strTextView =
        [
                ["Ａ","Ｂ","Ｃ","Ｄ","Ｅ","Ｆ","Ｇ","Ｈ","Ｉ","Ｊ","Ｋ","Ｌ","
Ｍ","Ｎ","Ｏ","Ｐ"],
                ["Ｑ","Ｒ","Ｓ","Ｔ","Ｕ","Ｖ","Ｗ","Ｘ","Ｙ","Ｚ","．","，","
：","；","＿","＠"],
                ["ａ","ｂ","ｃ","ｄ","ｅ","ｆ","ｇ","ｈ","ｉ","ｊ","ｋ","ｌ","
ｍ","ｎ","ｏ","ｐ"],
                ["ｑ","ｒ","ｓ","ｔ","ｕ","ｖ","ｗ","ｘ","ｙ","ｚ","＋","－","
／","＊","＝","％"],
                ["０","１","２","３","４","５","６","７","８","９","０","！","
？","’","”","＄"],
                ["（","）","｛","｝","［","］","＜","＞","＆","＃","｜","～","
＾","　","　","Fin"]
        ];

        let cursorX = 0;
        let cursorY = 0;
        let maxCursorX = length(strTextIn[0]);
        let maxCursorY = length(strTextIn);

        task TMenuItem(let itemX, let itemY)
        {
                let objTitleText = ObjText_Create();
                ObjText_SetFontType(objTitleText, "AsakuraSlab");
                ObjText_SetText(objTitleText, "10");
                ObjText_SetFontSize(objTitleText, 25);
                ObjText_SetFontBold(objTitleText, true);
                ObjText_SetFontColorTop(objTitleText, 255, 102, 102);
                ObjText_SetFontColorBottom(objTitleText, 255, 255, 255);
                ObjText_SetFontBorderType(objTitleText, BORDER_FULL);
                ObjText_SetFontBorderColor(objTitleText, 51, 0, 0);
                ObjText_SetFontBorderWidth(objTitleText, 2);
                Obj_SetRenderPriorityI(objTitleText, 30);
                ObjRender_SetX(objTitleText, 385);
                ObjRender_SetY(objTitleText, 430);

                let objText = CreateTextObject(120 + itemX * 24, 200 +
itemY * 24, 22, strTextView[itemY][itemX]);
                let objSelect = CreateTextObject(120 + itemX * 24, 200 +
itemY * 24, 22, strTextView[itemY][itemX]);
                ObjRender_SetBlendType(objSelect, BLEND_ADD_RGB);

                while(menuMode == MENU_NAME_ENTRY)
                {
                        Obj_SetVisible(objSelect, itemX == cursorX &&
itemY == cursorY);
                        yield;
                }
                Obj_Delete(objText);
                Obj_Delete(objSelect);
                Obj_Delete(objTitleText);
        }
        ascent(let iY in 0..maxCursorY)
        {
                ascent(let iX in 0 .. maxCursorX)
                {
                        TMenuItem(iX, iY);
                }
        }

        while(GetVirtualKeyState(VK_OK) != KEY_FREE){yield;}

        let userName = "";
        let objName = CreateTextObject(160, 96, 28, "");
        task TNameCursor()
        {
                let objCursor = CreateTextObject(0, 96, 28, "_");
                while(menuMode == MENU_NAME_ENTRY)
                {
                        let nameLength = length(userName);
                        ObjRender_SetX(objCursor, 160 + nameLength * 17);
                        Obj_SetVisible(objCursor, nameLength < 8);
                        yield;
                }
                Obj_Delete(objCursor);
        }
        TNameCursor;

        let frameKeyHold = 0;
        while(menuMode == MENU_NAME_ENTRY)
        {

                if(GetVirtualKeyState(VK_OK) == KEY_PULL)
                {
                        let nameLength = length(userName);
                        if(cursorX == maxCursorX-1 && cursorY ==
    maxCursorY-1)
                        {
                                if(nameLength == 0)
                                {
                                        userName = "Sin Nombre";
                                }
                                else
                                {
                                        SaveReplay(replayIndex, userName);
                                        SetScriptResult(RESULT_END);
                                        CloseScript(GetOwnScriptID());
                                        return;
                                }
                        }
                        else if(nameLength < 8)
                        {
                                userName = userName ~
strTextIn[cursorY][cursorX];
                        }
                }
                if(GetVirtualKeyState(VK_CANCEL) == KEY_PULL)
                {
                        let nameLength = length(userName);
                        if(nameLength > 0)
                        {
                                userName = userName[0..nameLength-1];
                        }
                }
                ObjText_SetText(objName, userName);

                if(GetVirtualKeyState(VK_UP) == KEY_PUSH ||
GetVirtualKeyState(VK_UP) == KEY_HOLD)
                {
                        frameKeyHold++;
                        if(GetVirtualKeyState(VK_UP) == KEY_PUSH ||
                                frameKeyHold == 20 ||
                                 (frameKeyHold > 20 && (frameKeyHold % 10
== 0)))
                        {
                                cursorY--;
                        }
                }
                else if(GetVirtualKeyState(VK_DOWN) == KEY_PUSH ||
GetVirtualKeyState(VK_DOWN) == KEY_HOLD)
                {
                        frameKeyHold++;
                        if(GetVirtualKeyState(VK_DOWN) == KEY_PUSH ||
                                frameKeyHold == 20 ||
                                 (frameKeyHold > 20 && (frameKeyHold % 10
== 0)))
                        {
                                cursorY++;
                        }
                }
                else if(GetVirtualKeyState(VK_LEFT) == KEY_PUSH ||
GetVirtualKeyState(VK_LEFT) == KEY_HOLD)
                {
                        frameKeyHold++;
                        if(GetVirtualKeyState(VK_LEFT) == KEY_PUSH ||
                                frameKeyHold == 20 ||
                                 (frameKeyHold > 20 && (frameKeyHold % 10
== 0)))
                        {
                                cursorX--;
                        }
                }
                else if(GetVirtualKeyState(VK_RIGHT) == KEY_PUSH ||
GetVirtualKeyState(VK_RIGHT) == KEY_HOLD)
                {
                        frameKeyHold++;
                        if(GetVirtualKeyState(VK_RIGHT) == KEY_PUSH ||
                                frameKeyHold == 20 ||
                                 (frameKeyHold > 20 && (frameKeyHold % 10
== 0)))
                        {
                                cursorX++;
                        }
                }
                else
                {
                        frameKeyHold = 0;
                }

                if(cursorX < 0)
                {
                        cursorX = maxCursorX-1;
                }
                else if(cursorX >= maxCursorX)
                {
                        cursorX = 0;
                }

                if(cursorY < 0)
                {
                        cursorY = maxCursorY-1;
                }
                else if(cursorY >= maxCursorY)
                {
                        cursorY = 0;
                }

                yield;
        }
```

```
}
```

ShotConst.txt

```
local
{
	let current = GetCurrentScriptDirectory();
	let path = current ~ "ShotData.txt";
	LoadEnemyShotData(path);
}

// 粒弾 --------------------------------
let DS_BALL_SS_RED		= 1;
let DS_BALL_SS_ORANGE	= 2;
let DS_BALL_SS_YELLOW	= 3;
let DS_BALL_SS_GREEN	= 4;
let DS_BALL_SS_SKY		= 5;
let DS_BALL_SS_BLUE		= 6;
let DS_BALL_SS_PURPLE	= 7;
let DS_BALL_SS_WHITE	= 8;

// 小弾 --------------------------------
let DS_BALL_S_RED		= 9;
let DS_BALL_S_ORANGE	= 10;
let DS_BALL_S_YELLOW	= 11;
let DS_BALL_S_GREEN		= 12;
let DS_BALL_S_SKY		= 13;
let DS_BALL_S_BLUE		= 14;
let DS_BALL_S_PURPLE	= 15;
let DS_BALL_S_WHITE		= 16;

// 小弾（加算合成描画用） --------------
let DS_BALL_S_A_RED		= 241;
let DS_BALL_S_A_ORANGE = 242;
let DS_BALL_S_A_YELLOW = 243;
let DS_BALL_S_A_GREEN  = 244;
let DS_BALL_S_A_SKY		= 245;
let DS_BALL_S_A_BLUE	= 246;
let DS_BALL_S_A_PURPLE = 247;
let DS_BALL_S_A_WHITE  = 248;

// 枠小弾 ------------------------------
let DS_BALL_BS_RED		= 17;
let DS_BALL_BS_ORANGE	= 18;
let DS_BALL_BS_YELLOW	= 19;
let DS_BALL_BS_GREEN	= 20;
let DS_BALL_BS_SKY             = 21;
let DS_BALL_BS_BLUE            = 22;
let DS_BALL_BS_PURPLE  = 23;
let DS_BALL_BS_WHITE   = 24;

// 中弾 --------------------------------
let DS_BALL_M_RED              = 25;
let DS_BALL_M_ORANGE   = 26;
let DS_BALL_M_YELLOW   = 27;
let DS_BALL_M_GREEN            = 28;
let DS_BALL_M_SKY              = 29;
let DS_BALL_M_BLUE             = 30;
let DS_BALL_M_PURPLE   = 31;
let DS_BALL_M_WHITE            = 32;

// 中弾（加算合成描画用） --------------
let DS_BALL_M_A_RED            = 225;
let DS_BALL_M_A_ORANGE = 226;
let DS_BALL_M_A_YELLOW = 227;
let DS_BALL_M_A_GREEN  = 228;
let DS_BALL_M_A_SKY            = 229;
let DS_BALL_M_A_BLUE   = 230;
let DS_BALL_M_A_PURPLE = 231;
let DS_BALL_M_A_WHITE  = 232;

// 針弾 --------------------------------
let DS_NEEDLE_RED              = 33;
let DS_NEEDLE_ORANGE   = 34;
let DS_NEEDLE_YELLOW   = 35;
let DS_NEEDLE_GREEN            = 36;
let DS_NEEDLE_SKY              = 37;
let DS_NEEDLE_BLUE             = 38;
let DS_NEEDLE_PURPLE   = 39;
let DS_NEEDLE_WHITE            = 40;

// 米粒弾 ------------------------------
let DS_RICE_S_RED              = 41;
let DS_RICE_S_ORANGE   = 42;
let DS_RICE_S_YELLOW   = 43;
let DS_RICE_S_GREEN            = 44;
let DS_RICE_S_SKY              = 45;
let DS_RICE_S_BLUE             = 46;
let DS_RICE_S_PURPLE   = 47;
let DS_RICE_S_WHITE            = 48;
```

```
// 氷塊弾 ------------------------------
let DS_ICE_RED          = 49;
let DS_ICE_ORANGE       = 50;
let DS_ICE_YELLOW       = 51;
let DS_ICE_GREEN        = 52;
let DS_ICE_SKY          = 53;
let DS_ICE_BLUE         = 54;
let DS_ICE_PURPLE       = 55;
let DS_ICE_WHITE        = 56;

// 座薬弾 ------------------------------
let DS_MISSILE_RED              = 57;
let DS_MISSILE_ORANGE   = 58;
let DS_MISSILE_YELLOW   = 59;
let DS_MISSILE_GREEN    = 60;
let DS_MISSILE_SKY              = 61;
let DS_MISSILE_BLUE             = 62;
let DS_MISSILE_PURPLE   = 63;
let DS_MISSILE_WHITE    = 64;

// 草履弾 ------------------------------
let DS_RICE_M_RED               = 65;
let DS_RICE_M_ORANGE    = 66;
let DS_RICE_M_YELLOW    = 67;
let DS_RICE_M_GREEN             = 68;
let DS_RICE_M_SKY               = 69;
let DS_RICE_M_BLUE              = 70;
let DS_RICE_M_PURPLE    = 71;
let DS_RICE_M_WHITE             = 72;

// 苦無弾 ------------------------------
let DS_KUNAI_RED                = 73;
let DS_KUNAI_ORANGE             = 74;
let DS_KUNAI_YELLOW             = 75;
let DS_KUNAI_GREEN              = 76;
let DS_KUNAI_SKY                = 77;
let DS_KUNAI_BLUE               = 78;
let DS_KUNAI_PURPLE             = 79;
let DS_KUNAI_WHITE              = 80;

// 鱗弾 --------------------------------
let DS_SCALE_RED                = 81;
let DS_SCALE_ORANGE             = 82;
let DS_SCALE_YELLOW             = 83;
let DS_SCALE_GREEN              = 84;
let DS_SCALE_SKY              = 85;
let DS_SCALE_BLUE             = 86;
let DS_SCALE_PURPLE           = 87;
let DS_SCALE_WHITE            = 88;

// 鱗弾（加算合成描画用） ---------------
let DS_SCALE_A_RED            = 233;
let DS_SCALE_A_ORANGE  = 234;
let DS_SCALE_A_YELLOW  = 235;
let DS_SCALE_A_GREEN   = 236;
let DS_SCALE_A_SKY            = 237;
let DS_SCALE_A_BLUE           = 238;
let DS_SCALE_A_PURPLE  = 239;
let DS_SCALE_A_WHITE   = 240;

// 札弾 --------------------------------
let DS_BILL_RED        = 89;
let DS_BILL_ORANGE     = 90;
let DS_BILL_YELLOW     = 91;
let DS_BILL_GREEN      = 92;
let DS_BILL_SKY        = 93;
let DS_BILL_BLUE       = 94;
let DS_BILL_PURPLE     = 95;
let DS_BILL_WHITE      = 96;

// 銭弾 --------------------------------
let DS_COIN_RED        = 97;
let DS_COIN_ORANGE     = 98;
let DS_COIN_YELLOW     = 99;
let DS_COIN_GREEN      = 100;
let DS_COIN_SKY        = 101;
let DS_COIN_BLUE       = 102;
let DS_COIN_PURPLE     = 103;
let DS_COIN_WHITE      = 104;

// 蝶弾 --------------------------------
let DS_BUTTERFLY_RED          = 105;
let DS_BUTTERFLY_ORANGE               = 106;
let DS_BUTTERFLY_YELLOW               = 107;
let DS_BUTTERFLY_GREEN        = 108;
let DS_BUTTERFLY_SKY          = 109;
let DS_BUTTERFLY_BLUE         = 110;
let DS_BUTTERFLY_PURPLE               = 111;
let DS_BUTTERFLY_WHITE        = 112;
```

```
// 光弹 --------------------------------
let DS_LIGHT_RED               = 113;
let DS_LIGHT_ORANGE            = 114;
let DS_LIGHT_YELLOW            = 115;
let DS_LIGHT_GREEN             = 116;
let DS_LIGHT_SKY               = 117;
let DS_LIGHT_BLUE              = 118;
let DS_LIGHT_PURPLE            = 119;
let DS_LIGHT_WHITE             = 120;

// 小星弹 ------------------------------
let DS_STAR_S_RED              = 121;
let DS_STAR_S_ORANGE   = 122;
let DS_STAR_S_YELLOW   = 123;
let DS_STAR_S_GREEN            = 124;
let DS_STAR_S_SKY              = 125;
let DS_STAR_S_BLUE             = 126;
let DS_STAR_S_PURPLE   = 127;
let DS_STAR_S_WHITE            = 128;

// 大星弹 ------------------------------
let DS_STAR_M_RED              = 129;
let DS_STAR_M_ORANGE   = 130;
let DS_STAR_M_YELLOW   = 131;
let DS_STAR_M_GREEN            = 132;
let DS_STAR_M_SKY              = 133;
let DS_STAR_M_BLUE             = 134;
let DS_STAR_M_PURPLE   = 135;
let DS_STAR_M_WHITE            = 136;

// 小刀弹 ------------------------------
let DS_KNIFE_YOUMU_RED         = 137;
let DS_KNIFE_YOUMU_ORANGE      = 138;
let DS_KNIFE_YOUMU_YELLOW      = 139;
let DS_KNIFE_YOUMU_GREEN       = 140;
let DS_KNIFE_YOUMU_SKY         = 141;
let DS_KNIFE_YOUMU_BLUE               = 142;
let DS_KNIFE_YOUMU_PURPLE      = 143;
let DS_KNIFE_YOUMU_WHITE       = 144;

// ナイフ弹 ----------------------------
let DS_KNIFE_KOUMA_RED         = 145;
let DS_KNIFE_KOUMA_ORANGE      = 146;
let DS_KNIFE_KOUMA_YELLOW      = 147;
let DS_KNIFE_KOUMA_GREEN       = 148;
let DS_KNIFE_KOUMA_SKY          = 149;
let DS_KNIFE_KOUMA_BLUE                 = 150;
let DS_KNIFE_KOUMA_PURPLE       = 151;
let DS_KNIFE_KOUMA_WHITE        = 152;

// 光線 --------------------------------
let DS_BEAM_RED                 = 153;
let DS_BEAM_ORANGE              = 154;
let DS_BEAM_YELLOW              = 155;
let DS_BEAM_GREEN               = 156;
let DS_BEAM_SKY                 = 157;
let DS_BEAM_BLUE                = 158;
let DS_BEAM_PURPLE              = 159;
let DS_BEAM_WHITE               = 160;
let DS_BEAM_RAINBOW             = 161;

// 炎弾 --------------------------------
let DS_FIRE_RED         = 162;
let DS_FIRE_BLUE        = 163;

// 卒塔婆弾 ----------------------------
let DS_TABLET   = 164;

// 大弾 --------------------------------
let DS_BALL_L_RED               = 165;
let DS_BALL_L_ORANGE    = 166;
let DS_BALL_L_YELLOW    = 167;
let DS_BALL_L_GREEN             = 168;
let DS_BALL_L_SKY               = 169;
let DS_BALL_L_BLUE              = 170;
let DS_BALL_L_PURPLE    = 171;
let DS_BALL_L_WHITE             = 172;

// 反転小弾 ----------------------------
let DS_BALL_S_R_WHITE  = 173;
let DS_BALL_S_R_RED             = 174;
let DS_BALL_S_R_PURPLE = 175;
let DS_BALL_S_R_BLUE   = 176;

// 反転枠小弾 --------------------------
let DS_BALL_BS_R_WHITE          = 177;
let DS_BALL_BS_R_RED            = 178;
let DS_BALL_BS_R_PURPLE                 = 179;
let DS_BALL_BS_R_BLUE           = 180;
```

```
// 反転中弾 ----------------------------
let DS_BALL_M_R_WHITE  = 181;
let DS_BALL_M_R_RED            = 182;
let DS_BALL_M_R_PURPLE = 183;
let DS_BALL_M_R_BLUE   = 184;

// 反転針弾 ----------------------------
let DS_NEEDLE_R_WHITE  = 185;
let DS_NEEDLE_R_RED            = 186;
let DS_NEEDLE_R_PURPLE = 187;
let DS_NEEDLE_R_BLUE   = 188;

// 反転米粒弾 --------------------------
let DS_RICE_S_R_WHITE  = 189;
let DS_RICE_S_R_RED            = 190;
let DS_RICE_S_R_PURPLE = 191;
let DS_RICE_S_R_BLUE   = 192;

// 反転氷塊弾 --------------------------
let DS_ICE_R_WHITE             = 193;
let DS_ICE_R_RED               = 194;
let DS_ICE_R_PURPLE            = 195;
let DS_ICE_R_BLUE              = 196;

// 反転座薬弾 --------------------------
let DS_MISSILE_R_WHITE         = 197;
let DS_MISSILE_R_RED           = 198;
let DS_MISSILE_R_PURPLE                = 199;
let DS_MISSILE_R_BLUE          = 200;

// 反転草履弾 --------------------------
let DS_RICE_M_R_WHITE  = 201;
let DS_RICE_M_R_RED            = 202;
let DS_RICE_M_R_PURPLE = 203;
let DS_RICE_M_R_BLUE   = 204;

// 反転蝶弾 ----------------------------
let DS_BUTTERFLY_R_WHITE       = 205;
let DS_BUTTERFLY_R_RED         = 206;
let DS_BUTTERFLY_R_PURPLE      = 207;
let DS_BUTTERFLY_R_BLUE                = 208;

// 反転光弾 ----------------------------
let DS_LIGHT_R_WHITE   = 209;
let DS_LIGHT_R_RED              = 210;
let DS_LIGHT_R_PURPLE  = 211;
let DS_LIGHT_R_BLUE             = 212;

// 反転小星弾 --------------------------
let DS_STAR_S_R_WHITE  = 213;
let DS_STAR_S_R_RED             = 214;
let DS_STAR_S_R_PURPLE = 215;
let DS_STAR_S_R_BLUE   = 216;

// 反転大星弾 --------------------------
let DS_STAR_M_R_WHITE  = 217;
let DS_STAR_M_R_RED             = 218;
let DS_STAR_M_R_PURPLE = 219;
let DS_STAR_M_R_BLUE   = 220;

// 反転光線 ----------------------------
let DS_BEAM_R_WHITE             = 221;
let DS_BEAM_R_RED               = 222;
let DS_BEAM_R_PURPLE   = 223;
let DS_BEAM_R_BLUE              = 224;

// 天狗弾 ------------------------------
let DS_TENGU    = 254;

// 弾幕裁判弾 --------------------------
let DS_JUDGMENT        = 255;
```

ShotData.txt

```
#UserShotData

shot_image = "script/UFO/system/img/Shot.png"
delay_rect = (209, 474, 240, 505)


// 粒弾 --------------------------------
ShotData{
        id = 1
        rect = ( 0, 0, 12, 12 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 2
        rect = ( 12, 0, 24, 12 )
        delay_color = ( 255, 127, 63 )
```

```
	}
	ShotData{
		id = 3
		rect = ( 24, 0, 36, 12 )
		delay_color = ( 255,255, 63 )
	}
	ShotData{
		id = 4
		rect = ( 36, 0, 48, 12 )
		delay_color = ( 63, 255, 63 )
	}
	ShotData{
		id = 5
		rect = ( 48, 0, 60, 12 )
		delay_color = ( 63, 255, 255 )
	}
	ShotData{
		id = 6
		rect = ( 60, 0, 72, 12 )
		delay_color = ( 63, 63, 255 )
	}
	ShotData{
		id = 7
		rect = ( 72, 0, 84, 12 )
		delay_color = ( 255, 63, 255 )
	}
	ShotData{
		id = 8
		rect = ( 84, 0, 96, 12 )
		delay_color = ( 255, 255, 255 )
	}

	// 小弾 --------------------------------
	ShotData{
		id = 9
		rect = ( 0, 12, 17, 29 )
		delay_color = ( 255, 63, 63 )
	}
	ShotData{
		id = 10
		rect = ( 18, 12, 35, 29 )
		delay_color = ( 255, 127, 63 )
	}
	ShotData{
		id = 11
```

```
        rect = ( 36, 12, 53, 29 )
        delay_color = ( 255, 255, 63 )
}
ShotData{
        id = 12
        rect = ( 54, 12, 71, 29 )
        delay_color = ( 63, 255, 63 )
}
ShotData{
        id = 13
        rect = ( 72, 12, 89, 29 )
        delay_color = ( 63, 255, 255 )
}
ShotData{
        id = 14
        rect = ( 90, 12, 107, 29 )
        delay_color = ( 63, 63, 255 )
}
ShotData{
        id = 15
        rect = ( 108, 12, 125, 29 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 16
        rect = ( 126, 12, 143, 29 )
        delay_color = ( 255, 255, 255 )
}

// 小弾（加算合成描画用） --------------
ShotData{
        id = 241
        rect = ( 256, 332, 273, 349 )
        render = ADD
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 242
        rect = ( 274, 332, 291, 349 )
        render = ADD
        delay_color = ( 255, 127, 63 )
}
ShotData{
        id = 243
        rect = ( 292, 332, 309, 349 )
```

```
		render = ADD
		delay_color = ( 255, 255, 63 )
	}
	ShotData{
		id = 244
		rect = ( 310, 332, 327, 349 )
		render = ADD
		delay_color = ( 63, 255, 63 )
	}
	ShotData{
		id = 245
		rect = ( 328, 332, 345, 349 )
		render = ADD
		delay_color = ( 63, 255, 255 )
	}
	ShotData{
		id = 246
		rect = ( 346, 332, 363, 349 )
		render = ADD
		delay_color = ( 63, 63, 255 )
	}
	ShotData{
		id = 247
		rect = ( 364, 332, 381, 349 )
		render = ADD
		delay_color = ( 255, 63, 255 )
	}
	ShotData{
		id = 248
		rect = ( 382, 332, 399, 349 )
		render = ADD
		delay_color = ( 255, 255, 255 )
	}

	// 枠小弾 ------------------------------
	ShotData{
		id = 17
		rect = ( 0, 30, 19, 49 )
		delay_color = ( 255, 63, 63 )
	}
	ShotData{
		id = 18
		rect = ( 20, 30, 39, 49 )
		delay_color = ( 255, 127, 63 )
	}
```

```
ShotData{
        id = 19
        rect = ( 40, 30, 59, 49 )
        delay_color = ( 255, 255, 63 )
}
ShotData{
        id = 20
        rect = ( 60, 30, 79, 49 )
        delay_color = ( 63, 255, 63 )
}
ShotData{
        id = 21
        rect = ( 80, 30, 99, 49 )
        delay_color = ( 63, 255, 255 )
}
ShotData{
        id = 22
        rect = ( 100, 30, 119, 49 )
        delay_color = ( 63, 63, 255 )
}
ShotData{
        id = 23
        rect = ( 120, 30, 139, 49 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 24
        rect = ( 140, 30, 159, 49 )
        delay_color = ( 255, 255, 255 )
}

// 中弾 --------------------------------
ShotData{
        id = 25
        rect = ( 0, 50, 29, 79 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 26
        rect = ( 30, 50, 59, 79 )
        delay_color = ( 255, 127, 63 )
}
ShotData{
        id = 27
        rect = ( 60, 50, 89, 79 )
```

```
		delay_color = ( 255, 255, 63 )
	}
	ShotData{
		id = 28
		rect = ( 90, 50, 119, 79 )
		delay_color = ( 63, 255, 63 )
	}
	ShotData{
		id = 29
		rect = ( 120, 50, 149, 79 )
		delay_color = ( 63, 255, 255 )
	}
	ShotData{
		id = 30
		rect = ( 150, 50, 179, 79 )
		delay_color = ( 63, 63, 255 )
	}
	ShotData{
		id = 31
		rect = ( 180, 50, 209, 79 )
		delay_color = ( 255, 63, 255 )
	}
	ShotData{
		id = 32
		rect = ( 210, 50, 239, 79 )
		delay_color = ( 255, 255, 255 )
	}

	// 中弾（加算合成描画用） ---------------
	ShotData{
		id = 225
		rect = ( 256, 302, 285, 331 )
		render = ADD
		delay_color = ( 255, 63, 63 )
	}
	ShotData{
		id = 226
		rect = ( 286, 302, 315, 331 )
		render = ADD
		delay_color = ( 255, 127, 63 )
	}
	ShotData{
		id = 227
		rect = ( 316, 302, 345, 331 )
		render = ADD
		delay_color = ( 255, 255, 63 )
	}
	ShotData{
		id = 228
		rect = ( 346, 302, 375, 331 )
		render = ADD
		delay_color = ( 63, 255, 63 )
	}
	ShotData{
		id = 229
		rect = ( 376, 302, 405, 331 )
		render = ADD
		delay_color = ( 63, 255, 255 )
	}
	ShotData{
		id = 230
		rect = ( 406, 302, 435, 331 )
		render = ADD
		delay_color = ( 63, 63, 255 )
	}
	ShotData{
		id = 231
		rect = ( 436, 302, 465, 331 )
		render = ADD
		delay_color = ( 255, 63, 255 )
	}
	ShotData{
		id = 232
		rect = ( 466, 302, 495, 331 )
		render = ADD
		delay_color = ( 255, 255, 255 )
	}

	// 針弾 --------------------------------
	ShotData{
		id = 33
		rect = ( 0, 80, 9, 100 )
		delay_color = ( 255, 63, 63 )
	}
	ShotData{
		id = 34
		rect = ( 10, 80, 19, 100 )
		delay_color = ( 255, 127, 63 )
	}
	ShotData{
		id = 35
		rect = ( 20, 80, 29, 100 )
		delay_color = ( 255, 255, 63 )
	}
	ShotData{
		id = 36
		rect = ( 30, 80, 39, 100 )
		delay_color = ( 63, 255, 63 )
	}
	ShotData{
		id = 37
		rect = ( 40, 80, 49, 100 )
		delay_color = ( 63, 255, 255 )
	}
	ShotData{
		id = 38
		rect = ( 50, 80, 59, 100 )
		delay_color = ( 63, 63, 255 )
	}
	ShotData{
		id = 39
		rect = ( 60, 80, 69, 100 )
		delay_color = ( 255, 63, 255 )
	}
	ShotData{
		id = 40
		rect = ( 70, 80, 79, 100 )
		delay_color = ( 255, 255, 255 )
	}

	// 米粒弾 ------------------------------
	ShotData{
		id = 41
		rect = ( 0, 100, 11, 118 )
		delay_color = ( 255, 63, 63 )
	}
	ShotData{
		id = 42
		rect = ( 12, 100, 23, 118 )
		delay_color = ( 255, 127, 63 )
	}
	ShotData{
		id = 43
		rect = ( 24, 100, 35, 118 )
		delay_color = ( 255, 255, 63 )
```

```
}
ShotData{
        id = 44
        rect = ( 36, 100, 47, 118 )
        delay_color = ( 63, 255, 63 )
}
ShotData{
        id = 45
        rect = ( 48, 100, 59, 118 )
        delay_color = ( 63, 255, 255 )
}
ShotData{
        id = 46
        rect = ( 60, 100, 71, 118 )
        delay_color = ( 63, 63, 255 )
}
ShotData{
        id = 47
        rect = ( 72, 100, 83, 118 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 48
        rect = ( 84, 100, 95, 118 )
        delay_color = ( 255, 255, 255 )
}

// 氷塊弾 ------------------------------
ShotData{
        id = 49
        rect = ( 0, 118, 11, 138 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 50
        rect = ( 12, 118, 23, 138 )
        delay_color = ( 255, 127, 63 )
}
ShotData{
        id = 51
        rect = ( 24, 118, 35, 138 )
        delay_color = ( 255, 255, 63 )
}
ShotData{
        id = 52
        rect = ( 36, 118, 47, 138 )
        delay_color = ( 63, 255, 63 )
}
ShotData{
        id = 53
        rect = ( 48, 118, 59, 138 )
        delay_color = ( 63, 255, 255 )
}
ShotData{
        id = 54
        rect = ( 60, 118, 71, 138 )
        delay_color = ( 63, 63, 255 )
}
ShotData{
        id = 55
        rect = ( 72, 118, 83, 138 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 56
        rect = ( 84, 118, 95, 138 )
        delay_color = ( 255, 255, 255 )
}

// 座薬弾 ------------------------------
ShotData{
        id = 57
        rect = ( 0, 138, 11, 158 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 58
        rect = ( 12, 138, 23, 158 )
        delay_color = ( 255, 127, 63 )
}
ShotData{
        id = 59
        rect = ( 24, 138, 35, 158 )
        delay_color = ( 255, 255, 63 )
}
ShotData{
        id = 60
        rect = ( 36, 138, 47, 158 )
        delay_color = ( 63, 255, 63 )
}
```

```
ShotData{
        id = 61
        rect = ( 48, 138, 59, 158 )
        delay_color = ( 63, 255, 255 )
}
ShotData{
        id = 62
        rect = ( 60, 138, 71, 158 )
        delay_color = ( 63, 63, 255 )
}
ShotData{
        id = 63
        rect = ( 72, 138, 83, 158 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 64
        rect = ( 84, 138, 93, 158 )
        delay_color = ( 255, 255, 255 )
}

// 草履弾 ------------------------------
ShotData{
        id = 65
        rect = ( 0, 158, 17, 186 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 66
        rect = ( 18, 158, 35, 186 )
        delay_color = ( 255, 127, 63 )
}
ShotData{
        id = 67
        rect = ( 36, 158, 53, 186 )
        delay_color = ( 255, 255, 63 )
}
ShotData{
        id = 68
        rect = ( 54, 158, 71, 186 )
        delay_color = ( 63, 255, 63 )
}
ShotData{
        id = 69
        rect = ( 72, 158, 89, 186 )
```

```
		delay_color = ( 63, 255, 255 )
	}
	ShotData{
		id = 70
		rect = ( 90, 158, 107, 186 )
		delay_color = ( 63, 63, 255 )
	}
	ShotData{
		id = 71
		rect = ( 108, 158, 125, 186 )
		delay_color = ( 255, 63, 255 )
	}
	ShotData{
		id = 72
		rect = ( 126, 158, 143, 186 )
		delay_color = ( 255, 255, 255 )
	}

	// 苦無弾 ------------------------------
	ShotData{
		id = 73
		rect = ( 0, 186, 13, 208 )
		delay_color = ( 255, 63, 63 )
	}
	ShotData{
		id = 74
		rect = ( 14, 186, 27, 208 )
		delay_color = ( 255, 127, 63 )
	}
	ShotData{
		id = 75
		rect = ( 28, 186, 41, 208 )
		delay_color = ( 255, 255, 63 )
	}
	ShotData{
		id = 76
		rect = ( 42, 186, 55, 208 )
		delay_color = ( 63, 255, 63 )
	}
	ShotData{
		id = 77
		rect = ( 56, 186, 69, 208 )
		delay_color = ( 63, 255, 255 )
	}
	ShotData{
		id = 78
		rect = ( 70, 186, 83, 208 )
		delay_color = ( 63, 63, 255 )
	}
	ShotData{
		id = 79
		rect = ( 84, 186, 97, 208 )
		delay_color = ( 255, 63, 255 )
	}
	ShotData{
		id = 80
		rect = ( 98, 186, 111, 208 )
		delay_color = ( 255, 255, 255 )
	}

	// 鱗弾 --------------------------------
	ShotData{
		id = 81
		rect = ( 0, 209, 17, 224 )
		delay_color = ( 255, 63, 63 )
	}
	ShotData{
		id = 82
		rect = ( 18, 209, 35, 224 )
		delay_color = ( 255, 127, 63 )
	}
	ShotData{
		id = 83
		rect = ( 36, 209, 53, 224 )
		delay_color = ( 255, 255, 63 )
	}
	ShotData{
		id = 84
		rect = ( 54, 209, 71, 224 )
		delay_color = ( 63, 255, 63 )
	}
	ShotData{
		id = 85
		rect = ( 72, 209, 89, 224 )
		delay_color = ( 63, 255, 255 )
	}
	ShotData{
		id = 86
		rect = ( 90, 209, 107, 224 )
		delay_color = ( 63, 63, 255 )
```

```
}
ShotData{
        id = 87
        rect = ( 108, 209, 125, 224 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 88
        rect = ( 126, 209, 143, 224 )
        delay_color = ( 255, 255, 255 )
}

// 鱗弾（加算合成描画用） ---------------
ShotData{
        id = 233
        rect = ( 0, 493, 17, 508 )
        render = ADD
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 234
        rect = ( 18, 493, 35, 508 )
        render = ADD
        delay_color = ( 255, 127, 63 )
}
ShotData{
        id = 235
        rect = ( 36, 493, 53, 508 )
        render = ADD
        delay_color = ( 255, 255, 63 )
}
ShotData{
        id = 236
        rect = ( 54, 493, 71, 508 )
        render = ADD
        delay_color = ( 63, 255, 63 )
}
ShotData{
        id = 237
        rect = ( 72, 493, 89, 508 )
        render = ADD
        delay_color = ( 63, 255, 255 )
}
ShotData{
        id = 238
        rect = ( 90, 493, 107, 508 )
        render = ADD
        delay_color = ( 63, 63, 255 )
    }
    ShotData{
        id = 239
        rect = ( 108, 493, 125, 508 )
        render = ADD
        delay_color = ( 255, 63, 255 )
    }
    ShotData{
        id = 240
        rect = ( 126, 493, 143, 508 )
        render = ADD
        delay_color = ( 255, 255, 255 )
    }

    // 札弾 --------------------------------
    ShotData{
        id = 89
        rect = ( 0, 224, 16, 242 )
        delay_color = ( 255, 63, 63 )
    }
    ShotData{
        id = 90
        rect = ( 16, 224, 32, 242 )
        delay_color = ( 255, 127, 63 )
    }
    ShotData{
        id = 91
        rect = ( 32, 224, 48, 242 )
        delay_color = ( 255, 255, 63 )
    }
    ShotData{
        id = 92
        rect = ( 48, 224, 64, 242 )
        delay_color = ( 63, 255, 63 )
    }
    ShotData{
        id = 93
        rect = ( 64, 224, 80, 242 )
        delay_color = ( 63, 255, 255 )
    }
    ShotData{
        id = 94
        rect = ( 80, 224, 96, 242 )
        delay_color = ( 63, 63, 255 )
}
ShotData{
        id = 95
        rect = ( 96, 224, 112, 242 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 96
        rect = ( 112, 224, 128, 242 )
        delay_color = ( 255, 255, 255 )
}

// 銭弾 --------------------------------
ShotData{
        id = 97
        rect = ( 0, 242, 19, 260 )
        delay_color = ( 255, 63, 63 )
        angular_velocity = 2.5
}
ShotData{
        id = 98
        rect = ( 20, 242, 39, 260 )
        delay_color = ( 255, 127, 63 )
        angular_velocity = 2.5
}
ShotData{
        id = 99
        rect = ( 40, 242, 59, 260 )
        delay_color = ( 255, 255, 63 )
        angular_velocity = 2.5
}
ShotData{
        id = 100
        rect = ( 60, 242, 79, 260 )
        delay_color = ( 63, 255, 63 )
        angular_velocity = 2.5
}
ShotData{
        id = 101
        rect = ( 80, 242, 99, 260 )
        delay_color = ( 63, 255, 255 )
        angular_velocity = 2.5
}
```

```
ShotData{
        id = 102
        rect = ( 100, 242, 119, 260 )
        delay_color = ( 63, 63, 255 )
        angular_velocity = 2.5
}
ShotData{
        id = 103
        rect = ( 120, 242, 139, 260 )
        delay_color = ( 255, 63, 255 )
        angular_velocity = 2.5
}
ShotData{
        id = 104
        rect = ( 140, 242, 159, 260 )
        delay_color = ( 255, 255, 255 )
        angular_velocity = 2.5
}

// 蝶弾 --------------------------------
ShotData{
        id = 105
        rect = ( 0, 260, 30, 290 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 106
        rect = ( 30, 260, 60, 290 )
        delay_color = ( 255, 127, 63 )
}
ShotData{
        id = 107
        rect = ( 60, 260, 90, 290 )
        delay_color = ( 255, 255, 63 )
}
ShotData{
        id = 108
        rect = ( 90, 260, 120, 290 )
        delay_color = ( 63, 255, 63 )
}
ShotData{
        id = 109
        rect = ( 120, 260, 150, 290 )
        delay_color = ( 63, 255, 255 )
}
```

```
ShotData{
        id = 110
        rect = ( 150, 260, 180, 290 )
        delay_color = ( 63, 63, 255 )
}
ShotData{
        id = 111
        rect = ( 180, 260, 210, 290 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 112
        rect = ( 210, 260, 240, 290 )
        delay_color = ( 255, 255, 255 )
}

// 光弾 --------------------------------
ShotData{
        id = 113
        rect = ( 0, 290, 27, 317 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 114
        rect = ( 28, 290, 55, 317 )
        delay_color = ( 255, 127, 63 )
}
ShotData{
        id = 115
        rect = ( 56, 290, 83, 317 )
        delay_color = ( 255, 255, 63 )
}
ShotData{
        id = 116
        rect = ( 84, 290, 111, 317 )
        delay_color = ( 63, 255, 63 )
}
ShotData{
        id = 117
        rect = ( 112, 290, 139, 317 )
        delay_color = ( 63, 255, 255 )
}
ShotData{
        id = 118
        rect = ( 140, 290, 167, 317 )
        delay_color = ( 63, 63, 255 )
}
ShotData{
        id = 119
        rect = ( 168, 290, 195, 317 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 120
        rect = ( 196, 290, 223, 317 )
        delay_color = ( 255, 255, 255 )
}

// 小星弹 ------------------------------
ShotData{
        id = 121
        rect = ( 0, 318, 19, 338 )
        delay_color = ( 255, 63, 63 )
        angular_velocity = -3
}
ShotData{
        id = 122
        rect = ( 20, 318, 39, 338 )
        delay_color = ( 255, 127, 63 )
        angular_velocity = -3
}
ShotData{
        id = 123
        rect = ( 40, 318, 59, 338 )
        delay_color = ( 255, 255, 63 )
        angular_velocity = -3
}
ShotData{
        id = 124
        rect = ( 60, 318, 79, 338 )
        delay_color = ( 63, 255, 63 )
        angular_velocity = -3
}
ShotData{
        id = 125
        rect = ( 80, 318, 99, 338 )
        delay_color = ( 63, 255, 255 )
        angular_velocity = -3
}
ShotData{
        id = 126
        rect = ( 100, 318, 119, 338 )
        delay_color = ( 63, 63, 255 )
        angular_velocity = -3
}
ShotData{
        id = 127
        rect = ( 120, 318, 139, 338 )
        delay_color = ( 255, 63, 255 )
        angular_velocity = -3
}
ShotData{
        id = 128
        rect = ( 140, 318, 159, 338 )
        delay_color = ( 255, 255, 255 )
        angular_velocity = -3
}

// 大星弹 ------------------------------
ShotData{
        id = 129
        rect = ( 0, 338, 32, 370 )
        delay_color = ( 255, 63, 63 )
        angular_velocity = 2
}
ShotData{
        id = 130
        rect = ( 32, 338, 64, 370 )
        delay_color = ( 255, 127, 63 )
        angular_velocity = 2
}
ShotData{
        id = 131
        rect = ( 64, 338, 96, 370 )
        delay_color = ( 255, 255, 63 )
        angular_velocity = 2
}
ShotData{
        id = 132
        rect = ( 96, 338, 128, 370 )
        delay_color = ( 63, 255, 63 )
        angular_velocity = 2
}
ShotData{
        id = 133
```

```
		rect = ( 128, 338, 160, 370 )
		delay_color = ( 63, 255, 255 )
		angular_velocity = 2
	}
	ShotData{
		id = 134
		rect = ( 160, 338, 192, 370 )
		delay_color = ( 63, 63, 255 )
		angular_velocity = 2
	}
	ShotData{
		id = 135
		rect = ( 192, 338, 224, 370 )
		delay_color = ( 255, 63, 255 )
		angular_velocity = 2
	}
	ShotData{
		id = 136
		rect = ( 225, 339, 256, 370 )
		delay_color = ( 255, 255, 255 )
		angular_velocity = 2
	}

	// 小刀弾 ------------------------------
	ShotData{
		id = 137
		rect = ( 0, 370, 22, 398 )
		delay_color = ( 255, 63, 63 )
	}
	ShotData{
		id = 138
		rect = ( 22, 370, 44, 398 )
		delay_color = ( 255, 127, 63 )
	}
	ShotData{
		id = 139
		rect = ( 44, 370, 66, 398 )
		delay_color = ( 255, 255, 63 )
	}
	ShotData{
		id = 140
		rect = ( 66, 370, 88, 398 )
		delay_color = ( 63, 255, 63 )
	}
	ShotData{
		id = 141
		rect = ( 88, 370, 110, 398 )
		delay_color = ( 63, 255, 255 )
	}
	ShotData{
		id = 142
		rect = ( 110, 370, 132, 398 )
		delay_color = ( 63, 63, 255 )
	}
	ShotData{
		id = 143
		rect = ( 132, 370, 154, 398 )
		delay_color = ( 255, 63, 255 )
	}
	ShotData{
		id = 144
		rect = ( 154, 370, 176, 398 )
		delay_color = ( 255, 255, 255 )
	}

	// ナイフ弾 ----------------------------
	ShotData{
		id = 145
		rect = ( 0, 398, 22, 430 )
		delay_color = ( 255, 63, 63 )
	}
	ShotData{
		id = 146
		rect = ( 22, 398, 44, 430 )
		delay_color = ( 255, 127, 63 )
	}
	ShotData{
		id = 147
		rect = ( 44, 398, 66, 430 )
		delay_color = ( 255, 255, 63 )
	}
	ShotData{
		id = 148
		rect = ( 66, 398, 88, 430 )
		delay_color = ( 63, 255, 63 )
	}
	ShotData{
		id = 149
		rect = ( 88, 398, 110, 430 )
		delay_color = ( 63, 255, 255 )
```

```
        }
        ShotData{
                id = 150
                rect = ( 110, 398, 132, 430 )
                delay_color = ( 63, 63, 255 )
        }
        ShotData{
                id = 151
                rect = ( 132, 398, 154, 430 )
                delay_color = ( 255, 63, 255 )
        }
        ShotData{
                id = 152
                rect = ( 154, 398, 176, 430 )
                delay_color = ( 255, 255, 255 )
        }

        // 光線 --------------------------------
        ShotData{
                id = 153
                rect = ( 1, 431, 25, 459 )
                delay_color = ( 255, 63, 63 )
        }
        ShotData{
                id = 154
                rect = ( 27, 431, 51, 459 )
                delay_color = ( 255, 127, 63 )
        }
        ShotData{
                id = 155
                rect = ( 53, 431, 77, 459 )
                delay_color = ( 255, 255, 63 )
        }
        ShotData{
                id = 156
                rect = ( 79, 431, 103, 459 )
                delay_color = ( 63, 255, 63 )
        }
        ShotData{
                id = 157
                rect = ( 105, 431, 129, 459 )
                delay_color = ( 63, 255, 255 )
        }
        ShotData{
                id = 158
```

```
		rect = ( 131, 431, 155, 459 )
		delay_color = ( 63, 63, 255 )
	}
	ShotData{
		id = 159
		rect = ( 157, 431, 181, 459 )
		delay_color = ( 255, 63, 255 )
	}
	ShotData{
		id = 160
		rect = ( 183, 431, 207, 459 )
		delay_color = ( 255, 255, 255 )
	}
	ShotData{
		id = 161
		rect = ( 209, 431, 233, 459 )
		delay_color = ( 255, 63, 63 )
	}

	// 炎弾 --------------------------------
	ShotData{
		id = 162
		delay_color = ( 255, 63, 63 )
		AnimationData{
			animation_data = ( 4, 0, 460, 20, 492 )
			animation_data = ( 4, 20, 460, 40, 492 )
			animation_data = ( 4, 40, 460, 60, 492 )
		}
	}
	ShotData{
		id = 163
		delay_color = ( 63, 63, 255 )
		AnimationData{
			animation_data = ( 4, 60, 460, 80, 492 )
			animation_data = ( 4, 80, 460, 100, 492 )
			animation_data = ( 4, 100, 460, 120, 492 )
		}
	}

	// 卒塔婆弾 ----------------------------
	ShotData{
		id = 164
		rect = ( 120, 460, 134, 492 )
		delay_color = ( 255, 127, 63 )
	}
```

```
// 大弾 --------------------------------
ShotData{
        id = 165
        rect = ( 320, 0, 384, 64 )
        render = ADD
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 166
        rect = ( 448, 64, 512, 128 )
        render = ADD
        delay_color = ( 255, 127, 63 )
}
ShotData{
        id = 167
        rect = ( 320, 64, 384, 128 )
        render = ADD
        delay_color = ( 255, 255, 63 )
}
ShotData{
        id = 168
        rect = ( 384, 0, 448, 64 )
        render = ADD
        delay_color = ( 63, 255, 63 )
}
ShotData{
        id = 169
        rect = ( 384, 64, 448, 128 )
        render = ADD
        delay_color = ( 63, 255, 255 )
}
ShotData{
        id = 170
        rect = ( 448, 0, 512, 64 )
        render = ADD
        delay_color = ( 63, 63, 255 )
}
ShotData{
        id = 171
        rect = ( 256, 64, 320, 128 )
        render = ADD
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 172
        rect = ( 256, 0, 320, 64 )
        render = ADD
        delay_color = ( 255, 255, 255 )
}

// 反転小弾 ---------------------------
ShotData{
        id = 173
        rect = ( 256, 128, 273, 145 )
        delay_color = ( 255, 255, 255 )
}
ShotData{
        id = 174
        rect = ( 274, 128, 291, 145 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 175
        rect = ( 292, 128, 309, 145 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 176
        rect = ( 310, 128, 327, 145 )
        delay_color = ( 63, 63, 255 )
}

// 反転枠小弾 -------------------------
ShotData{
        id = 177
        rect = ( 256, 146, 275, 165 )
        delay_color = ( 255, 255, 255 )
}
ShotData{
        id = 178
        rect = ( 276, 146, 295, 165 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 179
        rect = ( 296, 146, 315, 165 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 180
        rect = ( 316, 146, 335, 165 )
        delay_color = ( 63, 63, 255 )
}

// 反転中弾 ----------------------------
ShotData{
        id = 181
        rect = ( 256, 166, 285, 195 )
        delay_color = ( 255, 255, 255 )
}
ShotData{
        id = 182
        rect = ( 286, 166, 315, 195 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 183
        rect = ( 316, 166, 345, 195 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 184
        rect = ( 346, 166, 375, 195 )
        delay_color = ( 63, 63, 255 )
}

// 反転針弾 ----------------------------
ShotData{
        id = 185
        rect = ( 256, 196, 265, 216 )
        delay_color = ( 255, 255, 255 )
}
ShotData{
        id = 186
        rect = ( 266, 196, 275, 216 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 187
        rect = ( 276, 196, 285, 216 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 188
```

```
		rect = ( 286, 196, 295, 216 )
		delay_color = ( 63, 63, 255 )
	}

	// 反転米粒弾 --------------------------
	ShotData{
		id = 189
		rect = ( 256, 216, 267, 234 )
		delay_color = ( 255, 255, 255 )
	}
	ShotData{
		id = 190
		rect = ( 268, 216, 279, 234 )
		delay_color = ( 255, 63, 63 )
	}
	ShotData{
		id = 191
		rect = ( 280, 216, 291, 234 )
		delay_color = ( 255, 63, 255 )
	}
	ShotData{
		id = 192
		rect = ( 292, 216, 303, 234 )
		delay_color = ( 63, 63, 255 )
	}

	// 反転氷塊弾 --------------------------
	ShotData{
		id = 193
		rect = ( 256, 234, 267, 254 )
		delay_color = ( 255, 255, 255 )
	}
	ShotData{
		id = 194
		rect = ( 268, 234, 279, 254 )
		delay_color = ( 255, 63, 63 )
	}
	ShotData{
		id = 195
		rect = ( 280, 234, 291, 254 )
		delay_color = ( 255, 63, 255 )
	}
	ShotData{
		id = 196
		rect = ( 292, 234, 303, 254 )
```

```
        delay_color = ( 63, 63, 255 )
}

// 反転座薬弾 --------------------------
ShotData{
        id = 197
        rect = ( 256, 254, 267, 274 )
        delay_color = ( 255, 255, 255 )
}
ShotData{
        id = 198
        rect = ( 268, 254, 279, 274 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 199
        rect = ( 280, 254, 291, 274 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 200
        rect = ( 292, 254, 303, 274 )
        delay_color = ( 63, 63, 255 )
}

// 反転草履弾 --------------------------
ShotData{
        id = 201
        rect = ( 256, 274, 273, 302 )
        delay_color = ( 255, 255, 255 )
}
ShotData{
        id = 202
        rect = ( 274, 274, 291, 302 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 203
        rect = ( 292, 274, 309, 302 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 204
        rect = ( 310, 274, 327, 302 )
        delay_color = ( 63, 63, 255 )
```

```
}

// 反転蝶弾 ----------------------------
ShotData{
        id = 205
        rect = ( 384, 128, 414, 158 )
        delay_color = ( 255, 255, 255 )
}
ShotData{
        id = 206
        rect = ( 414, 128, 444, 158 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 207
        rect = ( 444, 128, 474, 158 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 208
        rect = ( 474, 128, 504, 158 )
        delay_color = ( 63, 63, 255 )
}

// 反転光弾 ----------------------------
ShotData{
        id = 209
        rect = ( 384, 158, 411, 185 )
        delay_color = ( 255, 255, 255 )
}
ShotData{
        id = 210
        rect = ( 412, 158, 439, 185 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 211
        rect = ( 440, 158, 467, 185 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 212
        rect = ( 468, 158, 495, 185 )
        delay_color = ( 63, 63, 255 )
}
```

```
// 反転小星弾 --------------------------
ShotData{
        id = 213
        rect = ( 384, 186, 403, 206 )
        delay_color = ( 255, 255, 255 )
        angular_velocity = -3
}
ShotData{
        id = 214
        rect = ( 404, 186, 423, 206 )
        delay_color = ( 255, 63, 63 )
        angular_velocity = -3
}
ShotData{
        id = 215
        rect = ( 424, 186, 443, 206 )
        delay_color = ( 255, 63, 255 )
        angular_velocity = -3
}
ShotData{
        id = 216
        rect = ( 444, 186, 463, 206 )
        delay_color = ( 63, 63, 255 )
        angular_velocity = -3
}

// 反転大星弾 --------------------------
ShotData{
        id = 217
        rect = ( 384, 206, 416, 238 )
        delay_color = ( 255, 255, 255 )
        angular_velocity = 2
}
ShotData{
        id = 218
        rect = ( 416, 206, 448, 238 )
        delay_color = ( 255, 63, 63 )
        angular_velocity = 2
}
ShotData{
        id = 219
        rect = ( 448, 206, 480, 238 )
        delay_color = ( 255, 63, 255 )
        angular_velocity = 2
}
ShotData{
        id = 220
        rect = ( 480, 206, 512, 238 )
        delay_color = ( 63, 63, 255 )
        angular_velocity = 2
}

// 反転光線 ----------------------------
ShotData{
        id = 221
        rect = ( 385, 239, 409, 267 )
        delay_color = ( 255, 255, 255 )
}
ShotData{
        id = 222
        rect = ( 411, 239, 435, 267 )
        delay_color = ( 255, 63, 63 )
}
ShotData{
        id = 223
        rect = ( 437, 239, 461, 267 )
        delay_color = ( 255, 63, 255 )
}
ShotData{
        id = 224
        rect = ( 463, 239, 487, 267 )
        delay_color = ( 63, 63, 255 )
}

// 天狗弾 ------------------------------
ShotData{
        id = 254
        rect = ( 480, 384, 511, 512 )
        render = ADD
        delay_color = ( 255, 255, 255 )
}

// 弾幕裁判弾 --------------------------
ShotData{
        id = 255
        rect = ( 385, 417, 479, 511 )
        delay_color = ( 255, 255, 255 )
        angular_velocity = 1
}
```

System.txt

```
let dirCurrent = GetCurrentScriptDirectory();
InstallFont(GetCurrentScriptDirectory() ~ "font/PintoLunaire.ttf");

@Initialize
{
        InitFrame();
        TScore();
        TGraze();
        TPlayerLife();
        TPlayerSpell();
        TBossLife();
        TBossTimer();
        TCurrentFps();
        TReplayFps();
}

@MainLoop
{
        yield;
}

@Event
{
        alternative(GetEventType())
        case(EV_START_BOSS_SPELL)
        {
                let path = dirCurrent ~ "System_MagicCircle.txt";
                let id = LoadScript(path);
                StartScript(id);
        }
        case(EV_GAIN_SPELL)
        {
                let objScene = GetEnemyBossSceneObjectID();
                let score = ObjEnemyBossScene_GetInfo(objScene,
INFO_SPELL_SCORE);
                TGainSpell(score);
        }
}

function InitFrame()
{
        let path = GetCurrentScriptDirectory() ~
"img/SystemBackground.png";
        let obj = ObjPrim_Create(OBJ_SPRITE_2D);
        ObjPrim_SetTexture(obj, path);
        Obj_SetRenderPriority(obj, 0);
        ObjSprite2D_SetSourceRect(obj, 0, 0, SCREEN_WIDTH, SCREEN_HEIGHT);
        ObjSprite2D_SetDestRect(obj, 0, 0, SCREEN_WIDTH, SCREEN_HEIGHT);
}

task TScore()
{
        let objScore = ObjText_Create();
        ObjText_SetFontType(objScore, "Oswald-Regular");
        ObjText_SetText(objScore, "Marcador");
        ObjText_SetFontSize(objScore, 21);
        ObjText_SetFontBold(objScore, true);
        ObjText_SetFontColorTop(objScore, 255, 255, 255);
        ObjText_SetFontColorBottom(objScore, 255, 255, 255);
        ObjText_SetFontBorderType(objScore, BORDER_FULL);
        ObjText_SetFontBorderColor(objScore,0, 0, 0);
        ObjText_SetFontBorderWidth(objScore, 2);
        Obj_SetRenderPriority(objScore, 0.01);
        ObjRender_SetX(objScore, 428);
        ObjRender_SetY(objScore, 48);

        let pathDigit = GetCurrentScriptDirectory() ~
"img/SystemDigit.png";
        let count = 12;

        let obj = ObjPrim_Create(OBJ_SPRITE_LIST_2D);
        ObjPrim_SetTexture(obj, pathDigit);
        Obj_SetRenderPriority(obj, 0.1);
        ObjRender_SetY(obj, 72);

        while(true)
        {
                let score = GetScore();
                score = min(score, 999999999999);
                let listNum = DigitToArray(score, count);

                ObjSpriteList2D_ClearVertexCount(obj);
                ascent(iObj in 0 .. count)
                {
                        let num = listNum[iObj];
                        ObjRender_SetX(obj, 440 + iObj * 14);
                        ObjSpriteList2D_SetSourceRect(obj, num * 36, 0,
(num + 1) * 36, 32);
                        ObjSpriteList2D_SetDestRect(obj, 0, 0, 16, 24);
                        ObjSpriteList2D_AddVertex(obj);
```

```
			}
			yield;
		}
	}

	task TGraze()
	{
		let objGraze = ObjText_Create();
		ObjText_SetFontType(objGraze, "Oswald-Regular");
		ObjText_SetText(objGraze, "Graze");
		ObjText_SetFontSize(objGraze, 21);
		ObjText_SetFontBold(objGraze, true);
		ObjText_SetFontColorTop(objGraze, 255, 255, 255);
		ObjText_SetFontColorBottom(objGraze, 255, 255, 255);
		ObjText_SetFontBorderType(objGraze, BORDER_FULL);
		ObjText_SetFontBorderColor(objGraze,0, 0, 0);
		ObjText_SetFontBorderWidth(objGraze, 2);
		Obj_SetRenderPriority(objGraze, 0.01);
		ObjRender_SetX(objGraze, 428);
		ObjRender_SetY(objGraze, 98);

		let pathDigit = GetCurrentScriptDirectory() ~
	"img/SystemDigit.png";
		let count = 5;

		let obj = ObjPrim_Create(OBJ_SPRITE_LIST_2D);
		ObjPrim_SetTexture(obj, pathDigit);
		Obj_SetRenderPriority(obj, 0.1);
		ObjRender_SetY(obj, 122);

		while(true)
		{
			let graze = GetGraze();
			graze = min(graze, 99999);
			let listNum = DigitToArray(graze, count);

			ObjSpriteList2D_ClearVertexCount(obj);
			ascent(iObj in 0 .. count)
			{
				let num = listNum[iObj];
				ObjRender_SetX(obj, 440 + iObj * 14);
				ObjSpriteList2D_SetSourceRect(obj, num * 36, 0,
	(num + 1) * 36, 32);
				ObjSpriteList2D_SetDestRect(obj, 0, 0, 16, 24);
				ObjSpriteList2D_AddVertex(obj);
			}
```

```
                yield;
        }
}

task TPlayerLife
{
        let objText = ObjText_Create();
        ObjText_SetFontType(objText, "Oswald-Regular");
        ObjText_SetText(objText, "Vida");
        ObjText_SetFontSize(objText, 21);
        ObjText_SetFontBold(objText, true);
        ObjText_SetFontColorTop(objText, 204, 153, 255);
        ObjText_SetFontColorBottom(objText, 204, 153, 255);
        ObjText_SetFontBorderType(objText, BORDER_FULL);
        ObjText_SetFontBorderColor(objText,0, 0, 0);
        ObjText_SetFontBorderWidth(objText, 2);
        Obj_SetRenderPriority(objText, 0.01);
        ObjRender_SetX(objText, 428);
        ObjRender_SetY(objText, 150);

        let pathDigit = GetCurrentScriptDirectory() ~
"img/SystemDigit.png";
        let count = 2;

        let obj = ObjPrim_Create(OBJ_SPRITE_LIST_2D);
        ObjPrim_SetTexture(obj, pathDigit);
        Obj_SetRenderPriority(obj, 0.1);
        ObjRender_SetY(obj, 174);

        while(true)
        {
                let point = GetPlayerLife();
                point = min(point, 99);
                point = max(point, 0);
                let listNum = DigitToArray(point, count);

                ObjSpriteList2D_ClearVertexCount(obj);
                ascent(iObj in 0 .. count)
                {
                        let num = listNum[iObj];
                        ObjRender_SetX(obj, 440 + iObj * 14);
                        ObjSpriteList2D_SetSourceRect(obj, num * 36, 0,
(num + 1) * 36, 32);
                        ObjSpriteList2D_SetDestRect(obj, 0, 0, 16, 24);
                        ObjSpriteList2D_AddVertex(obj);
                }
                yield;
        }
}

task TPlayerSpell
{
        let objText = ObjText_Create();
        ObjText_SetFontType(objText, "Oswald-Regular");
        ObjText_SetText(objText, "Hechizos");
        ObjText_SetFontSize(objText, 21);
        ObjText_SetFontBold(objText, true);
        ObjText_SetFontColorTop(objText, 102, 204, 0);
        ObjText_SetFontColorBottom(objText, 102, 204, 0);
        ObjText_SetFontBorderType(objText, BORDER_FULL);
        ObjText_SetFontBorderColor(objText,0, 0, 0);
        ObjText_SetFontBorderWidth(objText, 2);
        Obj_SetRenderPriority(objText, 0.01);
        ObjRender_SetX(objText, 428);
        ObjRender_SetY(objText, 202);

        let pathDigit = GetCurrentScriptDirectory() ~
"img/SystemDigit.png";
        let count = 2;

        let obj = ObjPrim_Create(OBJ_SPRITE_LIST_2D);
        ObjPrim_SetTexture(obj, pathDigit);
        Obj_SetRenderPriority(obj, 0.1);
        ObjRender_SetY(obj, 226);

        while(true)
        {
                let point = GetPlayerSpell();
                point = min(point, 99);
                let listNum = DigitToArray(point, count);

                ObjSpriteList2D_ClearVertexCount(obj);
                ascent(iObj in 0 .. count)
                {
                        let num = listNum[iObj];
                        ObjRender_SetX(obj, 440 + iObj * 14);
                        ObjSpriteList2D_SetSourceRect(obj, num * 36, 0,
(num + 1) * 36, 32);
                        ObjSpriteList2D_SetDestRect(obj, 0, 0, 16, 24);
                        ObjSpriteList2D_AddVertex(obj);
                }
                yield;
        }
}

task TBossLife
{
        let path = GetCurrentScriptDirectory() ~ "img/System.png";
        let obj = ObjPrim_Create(OBJ_SPRITE_LIST_2D);
        ObjPrim_SetTexture(obj, path);
        Obj_SetRenderPriority(obj, 0.7);

        let lastRemStep = -1;
        let lifeRateRender = 0;

        let objScene = ID_INVALID;
        loop
        {
                objScene = GetEnemyBossSceneObjectID();
                ObjSpriteList2D_ClearVertexCount(obj);
                if(objScene != ID_INVALID)
                {
                        RenderLife();
                }
                yield;
        }


        function RenderLife()
        {
                let countRemStep = ObjEnemyBossScene_GetInfo(objScene,
INFO_REMAIN_STEP_COUNT);
                if(lastRemStep != countRemStep)
                {
                        lifeRateRender = 0;
                }

                let lifeTotalMax = ObjEnemyBossScene_GetInfo(objScene,
INFO_ACTIVE_STEP_TOTAL_MAX_LIFE);
                let lifeTotal = ObjEnemyBossScene_GetInfo(objScene,
INFO_ACTIVE_STEP_TOTAL_LIFE);
                let lifeRate = min(lifeTotal / lifeTotalMax,
lifeRateRender);
                ObjSpriteList2D_SetSourceRect(obj, 1, 1, 127, 11);
                ObjSpriteList2D_SetDestRect(obj, 72, 8, 72 + 270 *
lifeRate, 12);
                ObjSpriteList2D_AddVertex(obj);
                ObjSpriteList2D_SetSourceRect(obj, 132, 1, 137, 11);
                let listLifeDiv = [0] ~
ObjEnemyBossScene_GetInfo(objScene, INFO_ACTIVE_STEP_LIFE_RATE_LIST);
                ascent(iDiv in 0 .. length(listLifeDiv))
                {
                        let rate = listLifeDiv[iDiv];
                        let x = 72 + 270 * (1-rate);
                        ObjSpriteList2D_SetDestRect(obj, x-1, 4, x + 1,
14);
                        ObjSpriteList2D_AddVertex(obj);
                }

                ObjSpriteList2D_SetSourceRect(obj, 1, 1, 127, 11);
                ascent(iStep in 0 .. countRemStep)
                {
                        let remStepRate = 58 / countRemStep;
                        ObjSpriteList2D_SetDestRect(obj, 4 + iStep *
remStepRate + 2, 8,
                                4 + (iStep + 1) * remStepRate, 12);
                        ObjSpriteList2D_AddVertex(obj);
                }

                lifeRateRender += 0.01;
                lifeRateRender = min(lifeRateRender, 1);
                lastRemStep = countRemStep;
        }
}

task TBossTimer
{
        let pathDigit = GetCurrentScriptDirectory() ~
"img/SystemDigit.png";

        let obj = ObjPrim_Create(OBJ_SPRITE_LIST_2D);
        ObjPrim_SetTexture(obj, pathDigit);
        Obj_SetRenderPriority(obj, 0.75);
        ObjRender_SetY(obj, 0);
        let count = 2;

        let objScene = ID_INVALID;
        loop
        {
                objScene = GetEnemyBossSceneObjectID();
                ObjSpriteList2D_ClearVertexCount(obj);
                if(objScene != ID_INVALID)
                {
                        RenderTimer();
                }
                yield;
        }

        function RenderTimer()
        {
                let timer = ObjEnemyBossScene_GetInfo(objScene,
INFO_TIMER);
                timer = min(timer, 99);
                let listNum = DigitToArray(timer, count);

                ObjSpriteList2D_ClearVertexCount(obj);
                ascent(iObj in 0 .. count)
                {
                        let num = listNum[iObj];
                        ObjRender_SetX(obj, 352 + iObj * 14);
                        ObjSpriteList2D_SetSourceRect(obj, num * 36, 0,
(num + 1) * 36, 32);
                        ObjSpriteList2D_SetDestRect(obj, 0, 0, 16, 24);
                        ObjSpriteList2D_AddVertex(obj);
                }
        }

}

task TGainSpell(score)
{
        let objText = ObjText_Create();
        ObjText_SetText(objText, "Bonus de Hechizo!");
        ObjText_SetFontSize(objText, 32);
        ObjText_SetFontBold(objText, true);
        ObjText_SetFontColorTop(objText, 255, 255, 255);
        ObjText_SetFontColorBottom(objText, 128, 128, 255);
        ObjText_SetFontBorderType(objText, BORDER_FULL);
        ObjText_SetFontBorderColor(objText,255, 255, 255);
        ObjText_SetFontBorderWidth(objText, 1);
        Obj_SetRenderPriority(objText, 0.6);
        ObjRender_SetX(objText, 32);
        ObjRender_SetY(objText, 98);

        let strScore = "+" ~ IntToString(score);
        let objScore = ObjText_Create();
        ObjText_SetText(objScore, strScore);
        ObjText_SetFontSize(objScore, 32);
        ObjText_SetFontBold(objScore, true);
```

```
		ObjText_SetFontColorTop(objScore, 255, 255, 255);
		ObjText_SetFontColorBottom(objScore, 255, 128, 128);
		ObjText_SetFontBorderType(objScore, BORDER_FULL);
		ObjText_SetFontBorderColor(objScore,255, 255, 255);
		ObjText_SetFontBorderWidth(objScore, 1);
		Obj_SetRenderPriority(objScore, 0.6);
		ObjRender_SetX(objScore, 180);
		ObjRender_SetY(objScore, 140);

		loop(120)
		{
			yield;
		}
		Obj_Delete(objText);
		Obj_Delete(objScore);
	}

	task TCurrentFps()
	{
		let objText = ObjText_Create();
		ObjText_SetFontSize(objText, 14);
		ObjText_SetFontBold(objText, true);
		ObjText_SetFontColorTop(objText, 204, 0, 102);
		ObjText_SetFontColorBottom(objText, 204, 0, 102);
		ObjText_SetFontBorderType(objText, BORDER_FULL);
		ObjText_SetFontBorderColor(objText, 255, 255, 255);
		ObjText_SetFontBorderWidth(objText, 2);
		ObjText_SetHorizontalAlignment(objText, ALIGNMENT_RIGHT);
		ObjText_SetMaxWidth(objText, GetScreenWidth() - 8);
		Obj_SetRenderPriority(objText, 1.0);
		ObjRender_SetX(objText, 0);
		ObjRender_SetY(objText, GetScreenHeight() - 20);

		loop
		{
			let fps = GetCurrentFps();
			let text = vtos("1.2f", fps) ~ "fps";
			ObjText_SetText(objText, text);
			yield;
		}
	}

	task TReplayFps()
	{
		if(!IsReplay()){return;}
```

```
        let objText = ObjText_Create();
        ObjText_SetFontSize(objText, 12);
        ObjText_SetFontBold(objText, true);
        ObjText_SetFontColorTop(objText, 128, 128, 255);
        ObjText_SetFontColorBottom(objText, 64, 64, 255);
        ObjText_SetFontBorderType(objText, BORDER_FULL);
        ObjText_SetFontBorderColor(objText,255, 255, 255);
        ObjText_SetFontBorderWidth(objText, 1);
        Obj_SetRenderPriority(objText, 1.0);

        let px = GetStgFrameLeft() + GetStgFrameWidth() - 18;
        let py = GetStgFrameTop() + GetScreenHeight() - 14;
        ObjRender_SetX(objText, px);
        ObjRender_SetY(objText, py);

        loop
        {
                let fps = GetReplayFps();
                let text = vtos("02d", fps);
                ObjText_SetText(objText, text);
                yield;
        }
}

function DigitToArray(let digit,let count)
{
        let res = [];
        digit = truncate(digit);

        loop
        {
                let tnum = truncate(digit % 10);
                digit /= 10;
                res = [tnum] ~ res;
                if(truncate(digit) == 0){break;}
        }

        loop(max(0, count - length(res)))
        {
                res = [0] ~ res;
        }

        return res;
}
```

System_MagicCircle.txt

```
let dirCurrent = GetCurrentScriptDirectory();
let typeEnd = 0;
let END_FAILED = 1;
let END_SUCCESS = 2;

@Initialize
{
        MagicCircle();
}

@MainLoop
{
        yield;
}

@Event
{
        alternative(GetEventType())
        case(EV_END_BOSS_STEP)
        {
                if(typeEnd == 0)
                {
                        typeEnd = END_FAILED;
                }
        }
        case(EV_GAIN_SPELL)
        {
                typeEnd = END_SUCCESS;
        }
}

task MagicCircle()
{
        let countVertex = 64;
        let listRadius = [];
        loop(countVertex)
        {
                listRadius = listRadius ~ [0];
        }

        let path = dirCurrent ~ "img/MagicCircle.png";
        let obj = ObjPrim_Create(OBJ_PRIMITIVE_2D);
        ObjPrim_SetPrimitiveType(obj, PRIMITIVE_TRIANGLESTRIP);
        ObjPrim_SetVertexCount(obj, countVertex);
        ObjRender_SetBlendType(obj, BLEND_ADD_RGB);
        Obj_SetRenderPriority(obj, 0.3);
        ObjPrim_SetTexture(obj, path);
        ascent(iVert in 0..countVertex / 2)
        {
                let left = iVert * 128;
                let indexVert = iVert * 2;
                ObjPrim_SetVertexUVT(obj, indexVert + 0, left, 0);
                ObjPrim_SetVertexUVT(obj, indexVert + 1, left, 64);

        }

        let objScene = GetEnemyBossSceneObjectID();
        let objBoss = GetEnemyBossObjectID()[0];
        let timerOrg = ObjEnemyBossScene_GetInfo(objScene,
INFO_ORGTIMERF);
        let bLastSpell = ObjEnemyBossScene_GetInfo(objScene,
INFO_IS_LAST_SPELL);

        let cx = 0;
        let cy = 0;
        let maxRadius = 256 * 1.2;
        let alpha = 0;
        let frame = 0;
        let angleRender = 0;

        function GetPlayerX()
        {
                let objPlayer = GetPlayerObjectID();
                return ObjRender_GetX(objPlayer);
        }

        function GetPlayerY()
        {
                let objPlayer = GetPlayerObjectID();
                return ObjRender_GetY(objPlayer);
        }

        function UpdateVertex()
        {
                if(bLastSpell)
                {
                        ObjRender_SetColor(obj, 255 * alpha, 192 * alpha,
192 * alpha);
                }
                else
                {
                        ObjRender_SetColor(obj, 192 * alpha, 192 * alpha,
255 * alpha);
                }

                ObjRender_SetPosition(obj, cx, cy, 0);
                ObjRender_SetAngleZ(obj, angleRender);
        }

        let pathUseSpell = dirCurrent ~ "system/se/seUseSpellCard.wav";
        LoadSound(pathUseSpell);
        PlaySE(pathUseSpell);

        while(typeEnd == 0)
        {
                if(!Obj_IsDeleted(objBoss))
                {
                        cx = ObjRender_GetX(objBoss);
                        cy = ObjRender_GetY(objBoss);
                }

                alpha += 1 / 120;
                alpha = min(alpha, 1);
                angleRender += 360 / countVertex / 4;

                let timer = ObjEnemyBossScene_GetInfo(objScene,
INFO_TIMERF);
                let rRate = timer / timerOrg;
                let bMiss = ObjEnemyBossScene_GetInfo(objScene,
INFO_PLAYER_SHOOTDOWN_COUNT) > 0 ||

ObjEnemyBossScene_GetInfo(objScene, INFO_PLAYER_SPELL_COUNT) > 0 ;

                ascent(iVert in 0..countVertex / 2)
                {
                        let indexVert = iVert * 2;
                        let angle = 360 / (countVertex / 2 - 1) * iVert;

                        let vx1 = listRadius[indexVert] * cos(angle);
                        let vy1 = listRadius[indexVert] * sin(angle);
                        ObjPrim_SetVertexPosition(obj, indexVert + 0, vx1,
vy1, 0);

                        let vx2 = listRadius[indexVert+1] * cos(angle);
                        let vy2 = listRadius[indexVert+1] * sin(angle);
                        ObjPrim_SetVertexPosition(obj, indexVert + 1, vx2,
vy2, 0);

                        if(frame >= 0)
                        {
                                let dr = (maxRadius * rRate -
listRadius[indexVert]) / 16;
                                listRadius[indexVert] =
listRadius[indexVert] + dr;
                        }
                        if(frame > 45)
                        {
                                let rRateIn = rRate - 0.08;
                                if(bMiss)
                                {
                                        rRateIn = rRate - 0.04;
                                }
                                if(rRateIn < 0){rRateIn=0;}
                                let dr= (maxRadius * rRateIn -
listRadius[indexVert + 1]) / 64;
                                listRadius[indexVert + 1] =
listRadius[indexVert + 1] + dr;
                        }

                }

                UpdateVertex();
                frame++;

                yield;
        }

        if(typeEnd == END_FAILED)
        {
                Obj_Delete(obj);
                CloseScript(GetOwnScriptID());
                return;
        }

        let pathGainSpell = dirCurrent ~
"system/se/seGetSpellCardBonus.wav";
        LoadSound(pathGainSpell);
        PlaySE(pathGainSpell);

        let rRate = 1.0;
        frame = 0;
        alpha = 1;
        loop(105)
        {
                angleRender += 360 / countVertex / 4;
                let dx = (GetPlayerX() - cx) / 16;
                let dy = (GetPlayerY() - cy) / 16;
                cx += dx;
                cy += dy;
                if(frame >= 45)
                {
                        alpha -= 1 / 45;
                        alpha = max(alpha, 0);
                }

                ascent(iVert in 0..countVertex / 2)
                {
                        let indexVert = iVert * 2;
                        let angle = 360 / (countVertex / 2 - 1) * iVert;

                        let vx1 = listRadius[indexVert] * cos(angle);
                        let vy1 = listRadius[indexVert] * sin(angle);
                        ObjPrim_SetVertexPosition(obj, indexVert + 0, vx1,
vy1, 0);

                        let vx2 = listRadius[indexVert+1] * cos(angle);
                        let vy2 = listRadius[indexVert+1] * sin(angle);
                        ObjPrim_SetVertexPosition(obj, indexVert + 1, vx2,
vy2, 0);

                        let drOut = 0;
                        let drIn = 0;
                        if(frame <= 45)
                        {
                                let rRateOut = 1.0;
                                drOut = (maxRadius * rRateOut -
listRadius[indexVert]) / 8;

                                let rRateIn = rRateOut - 0.08;
                                if(rRateIn<0){rRateIn=0;}
                                drIn = (maxRadius * rRateIn -
listRadius[indexVert+1]) / 8;
                        }
                        else
                        {
                                cx = GetPlayerX();
                                cy = GetPlayerY();
                                rRate -= 1.0 / 60.0;
                                let rRateOut = rRate * sin(angle % 60);
                                drOut = (maxRadius * rRateOut -
listRadius[indexVert]) / 16;

                                let rRateIn = rRate * sin(angle % 60)-0.08;
                                if(rRateIn<0){rRateIn=0;}
                                drIn=(maxRadius * rRateIn -
listRadius[indexVert+1])/16;
                        }
                        listRadius[indexVert] = listRadius[indexVert] +
drOut;
                        listRadius[indexVert + 1] = listRadius[indexVert +
1] + drIn;
                }

                UpdateVertex();
                frame++;
                yield;
        }

        Obj_Delete(obj);
        CloseScript(GetOwnScriptID());

}
```

Background.txt

```
let CSD = GetCurrentScriptDirectory;
let objScene = GetEnemyBossSceneObjectID;

#include".\PrimitiveTest.dnh"

let OnSpell = false;

@Initialize{
        SetCameraFocusX(0);
        SetCameraFocusY(0);
        SetCameraFocusZ(0);
        SetCameraRadius(1200);
    SetCameraElevationAngle(-10);
        SetCameraAzimuthAngle(90);

        Background;
        SpellBG;
}
```

```
@MainLoop{
    OnSpell = (GetEnemyBossSceneObjectID != ID_INVALID &&
ObjEnemyBossScene_GetInfo(GetEnemyBossSceneObjectID, INFO_IS_SPELL));
        yield;
}

function SpriteCreate{
let obj = ObjPrim_Create(OBJ_SPRITE_3D);
ObjPrim_SetTexture(obj,CSD~"./img/stgbg2.png");
Obj_SetRenderPriorityI(obj,20);
ObjRender_SetAngleX(obj,-50);
ObjSprite3D_SetSourceDestRect(obj,0,0,512,512);
ObjRender_SetScaleXYZ(obj,1,1,1.5);
ObjRender_SetBlendType(obj,BLEND_ALPHA);
ObjRender_SetColor(obj,150,150,150);
return obj
}

function SpriteCreate2{
let obj = ObjPrim_Create(OBJ_SPRITE_3D);
ObjPrim_SetTexture(obj,CSD~"./img/stgbg3.png");
Obj_SetRenderPriorityI(obj,20);
ObjRender_SetAngleX(obj,-50);
ObjSprite3D_SetSourceDestRect(obj,0,0,512,512);
ObjRender_SetScaleXYZ(obj,1,1,1.5);
ObjRender_SetColor(obj,150,150,150);
ObjRender_SetBlendType(obj,BLEND_ADD_ARGB);
return obj
}

task Background{
SetCameraElevationAngle(-20);
Create2DObject(GetCurrentScriptDirectory~"./img/stgbg4.png",256,324,"ADD"
,22,5,120,0.75,255,255,255);

        let obj1 = ObjPrim_Create(OBJ_SPRITE_2D);
        ObjRender_SetBlendType(obj1, BLEND_ALPHA);
        Obj_SetRenderPriorityI(obj1, 21);
        ObjPrim_SetTexture(obj1,
GetCurrentScriptDirectory~"./img/stgbg1.png");
        ObjRender_SetScaleXYZ(obj1, 1.1, 0.7, 1);
        ObjSprite2D_SetSourceRect(obj1, 0, 0, 328, 220);
        ObjSprite2D_SetDestRect(obj1, -256, -256, 256, 256);
        ObjRender_SetPosition(obj1, GetStgFrameWidth/2,
GetStgFrameHeight/2-130, 0);
```

```
let obj = [SpriteCreate];
let obj2 = [SpriteCreate2];
ObjRender_SetAlpha(obj[0],255);
        let movel = 0;
        let movel2 = 0;
        let movel3 = 0;
        loop{

    movel+=0;
    movel2+=1.5;
        movel3+=2.5;

ObjSprite3D_SetSourceDestRect(obj[0],512,0+movel2,(512*6)+movel,(512*6)+m
ovel2);
        ObjSprite3D_SetSourceDestRect(obj2[0],512,0+movel3,(512*6)+movel,(
512*6)+movel3);
        yield;
        }
}

task SpellBG{
        let obj1 = ObjPrim_Create(OBJ_SPRITE_2D);
        ObjRender_SetBlendType(obj1, BLEND_ALPHA);
        Obj_SetRenderPriorityI(obj1, 23);
        ObjPrim_SetTexture(obj1,
GetCurrentScriptDirectory~"./img/splbg1.png");
        ObjRender_SetScaleXYZ(obj1, 1.4, 1.4, 1);
        ObjSprite2D_SetSourceRect(obj1, 0, 0, 512, 512);
        ObjSprite2D_SetDestRect(obj1, -256, -256, 256, 256);
        ObjRender_SetPosition(obj1, GetScreenWidth/2, GetScreenHeight/2,
0);

        Create2DObject2(GetCurrentScriptDirectory~"./img/splbg2.png",256,1
024,"SUB",24,GetStgFrameHeight/2-
120,GetCommonData("BGAlpha",0),2,100,100,255);

        let frame = 0;
        let alpha = 0;
        let alpha2 = 0;
        let angle = 0;
        let angle2 = 0;
        let rect = 800;
        let rect2 = 1000;
        let slide = 0;
        loop{
                if(GetCommonData("SpellCard",false)){
                        if(alpha < 255) { alpha += 255/90;}
                        SetCommonData("BGAlpha",alpha);}
                else {if(alpha>0){alpha -= 4;}
                SetCommonData("BGAlpha",alpha);}

                frame++;
                ObjRender_SetAlpha(obj1, alpha);
                //angle+=0.5;
                slide+=0.5;
                ObjSprite2D_SetSourceRect(obj1,0,0,600,40000+slide);
                ObjSprite2D_SetDestCenter(obj1);

        ObjRender_SetPosition(obj1,GetScreenWidth/2,GetScreenHeight/2-
(slide*1.5),0);
                rect+=1;
                //rect2++;
                yield;}
}
function wait(w) { loop(w) { yield; } }
```

Cutin.txt

```
let current = GetCurrentScriptDirectory;
let DEBUG = false;
let CutinDifficulty = "";
let SpellAttack_img = current~"SpellAttack.png";
let img_SpellAttack = current~"SpellAttackText.png"; // Spell Declare
let Alpha_HUD = [255, 255]; //top, bottom

let NAZRIN = "NAZRIN";
let BYAKUREN = "BYAKUREN";
let KANAKO = "KANAKO";
let MOKOU = "MOKOU";
let AYA = "AYA";
let YABUSAME = "LENEN";

let LENEN = "LENEN";

//If you have any questions or requests, send them to gtbot/TheGtbot (I
go by either on different places)
//Version 1.3


task ObjCutin_SetSpellcardS3(SpellName, R, G, B){
        let r = IntToString(R);
        let g = IntToString(G);
        let b = IntToString(B);
        let boss = GetEnemyBossObjectID[0];
        let spells = Obj_GetValueD(boss, "Spellcards", []);
        spells = spells~[[SpellName, r, g, b, "", ""]];
        Obj_SetValue(boss, "Spellcards", spells);
}

task ObjCutin_SetSpellcardS4(SpellName, image, cuttype, R, G, B){
        let r = IntToString(R);
        let g = IntToString(G);
        let b = IntToString(B);
        let boss = GetEnemyBossObjectID[0];
        let spells = Obj_GetValueD(boss, "Spellcards", []);
        spells = spells~[[SpellName, r, g, b, cuttype, image]];
        Obj_SetValue(boss, "Spellcards", spells);
}

task ObjCutin_LaunchS3(type, image, difficulty){
        CutinS3(type, image, difficulty);
}

task CutinS3(Type, mimage, difficulty){
        let spells = Obj_GetValueD(GetEnemyBossObjectID[0], "Spellcards",
[]);
        let boss = GetEnemyBossObjectID[0];
        let spcount = length(spells);
        CutinDifficulty = difficulty;
        let IsSpellAttackAnimation = false;
        if(!IsCommonDataAreaExists("cutin_History")){
                CreateCommonDataArea("cutin_History");
        }
        LoadCommonDataAreaA1("cutin_History");


        //if(length(Images)>6)
        //      Images = [Images];
        ////    Colors = [Colors];
        //
        //let Colors = [[r, g, b]];

        descent(i in 0..spcount){SpellText(6*i, i);}
        SpawnCutinImage(Type, mimage);


        let delay = 6;
        while(!Obj_IsDeleted(boss)){
                spells = Obj_GetValueD(GetEnemyBossObjectID[0],
"Spellcards", []);
                if(length(spells)!=spcount&&delay==0){
                        delay = 6;
                        SpellText(0, spcount);
                        let newtype = spells[spcount][4];
                        if(newtype == ""){newtype = Type;}
                        SpawnCutinImage(newtype, spells[spcount][5]);
                        spcount++;
                }
                delay = max(delay-1, 0);
                yield;
        }



        task FireCutinA1(x, y, angle, ispeed, itime, mspeed, mtime, img){

                if(IsSpellAttackAnimation){return;}
                let SpellCutin = CreateSimple2DImageA1(0.29, img);
                ObjRender_SetPosition(SpellCutin, x, y, 0);
                ObjRender_SetAlpha(SpellCutin, 0);



                let len = itime;
                ascent(x in 0..len){
                        let mod = x/len;
                        let locs = [ObjRender_GetX(SpellCutin),
ObjRender_GetY(SpellCutin)];
                        ObjRender_SetPosition(SpellCutin,
floor(locs[0]+ispeed*cos(angle)), floor(locs[1]+ispeed*sin(angle)), 0);

                        let alph = min(255, 255*(mod*2));
                        ObjRender_SetAlpha(SpellCutin, alph);
                        yield;
                }

                len = mtime;
                let nx = ObjRender_GetX(SpellCutin);
                let ny = ObjRender_GetY(SpellCutin);
                ascent(x in 0..len){
                        nx += mspeed*cos(angle);
                        ny += mspeed*sin(angle);
                        ObjRender_SetPosition(SpellCutin, floor(nx),
floor(ny), 0);
```

```
                        yield;
                }
                //return;
                len = mtime;
                ascent(x in 0..len){
                        let mod = x/len;
                        let locs = [ObjRender_GetX(SpellCutin),
ObjRender_GetY(SpellCutin)];
                        ObjRender_SetPosition(SpellCutin,
floor(locs[0]+ispeed*cos(angle)), floor(locs[1]+ispeed*sin(angle)), 0);
                        yield;
                }
                Obj_Delete(SpellCutin);
        }

        task FireCutinA2(x, y, alpha, img, details){
                if(IsSpellAttackAnimation){return;}
                //[spd, spdinc, ang, anginc, alphachange, time]
                let SpellCutin = CreateSimple2DImageA1(0.29, img);
                ObjRender_SetPosition(SpellCutin, x, y, 0);
                ObjRender_SetAlpha(SpellCutin, alpha);

                ascent(i in 0..length(details)){
                        let nx = ObjRender_GetX(SpellCutin);
                        let ny = ObjRender_GetY(SpellCutin);
                        let info = details[i];
                        let len = info[5];
                        let alphinc = -(alpha-info[4])/len;
                        let spdinc = -(info[0]-info[1])/len;
                        let mspeed = info[0];
                        let cangle = info[2];
                        ascent(x in 0..len){
                                mspeed+=spdinc;
                                cangle+=info[3];
                                nx += mspeed*cos(cangle);
                                ny += mspeed*sin(cangle);
                                ObjRender_SetPosition(SpellCutin, nx, ny,
0);

                                yield;
                                alpha = min(max(alpha+alphinc, 0), 255);
                                ObjRender_SetAlpha(SpellCutin, alpha);
                        }
                }
                Obj_Delete(SpellCutin);
        }
        function SpawnCutinImage(type, image){
                //We use another "type" variable (lowercase t) so that
                //spellcards set after the root function is called will
play the
                //new set cutin type as opposed to the original one
                // same for image
                if(type == "NAZRIN"){
                        FireCutinA2(GetStgFrameWidth+208,
GetStgFrameHeight/2-144, 0, image,
                        [
                                [20.6, 20.6, 158, 0, 255, 20],
                                [0.65, 0.65, 158, 0, 255, 90],
                                [20.6, 20.6, 158, 0, 255, 20]
                        ]
                        );
                        SpellAttackEffect(145);
                }
                else if(type == "KANAKO"){
                        FireCutinA2(GetStgFrameWidth+180,
GetStgFrameHeight/2-190, 0, image,
                        [
                                [30, 8, 144, 0, 192, 23],
                                [8, 0, 144, 0, 255, 10],
                                [0, 2.1, 270, 0, 255, 90],
                                [2.1, 2.1, 270, 0, 255, 20],
                                [2.1, 1, 270, 0, 0, 20]
                        ]
                        );
                        SpellAttackEffect(145);
                }
                else if(type == "BYAKUREN"){
                        FireCutinA2(-256, GetStgFrameHeight/2, 0, image,
                        [
                                [14, 14, 0, 0, 255, 30],
                                [0.5, 0.5, 0, 0, 255, 90],
                                [14, 14, 0, 0, 0, 30]
                        ]
                        );
                        SpellAttackEffect(145);
                }
                else if(type == "LENEN" || type == "YABUSAME"){
                        FireCutinA2(GetStgFrameWidth+256,
GetStgFrameHeight/2+96, 0, image,
                        [
                                [32, 1, 182, 0.1, 255, 30],
                                [1, 1.25, 180, 14, 255, 10],
                                [1.25, 1.75, 315, 0, 255, 80],
                                [2, 2.5, 340, -15.75, 192, 10],
                                [2.5, 31, 182.5, 0, 0, 20],
                        ]
                        );
                        SpellAttackEffect(145);
                }
                else if(type == "AYA"){
                        SpellAttackEffect(145);
                }
                else if(type == "MOKOU"){
                }
                else{RaiseError("Not a valid cutin type.");}
        }

        task SpellText(delay, num){
                let tspell = spells[num];
                let SpellName = tspell[0];
                let r = atoi(tspell[1]);
                let g = atoi(tspell[2]);
                let b = atoi(tspell[3]);

                let Colors = [[r,g,b]];

                //These are long-looking common data group names
                let SpellDataAttempt =
SpellName~"|"~CutinDifficulty~"|"~GetPlayerID~"|"~"Attempt";
                let SpellDataGet =
SpellName~"|"~CutinDifficulty~"|"~GetPlayerID~"|"~"Get";
                //This is actually the common data retrieval
                let SpellValueAttempt = GetAreaCommonData("cutin_History",
SpellDataAttempt, 0);
                let SpellValueGet = GetAreaCommonData("cutin_History",
SpellDataGet, 0);


                if(!IsReplay&&!DEBUG&&num==0){
                        SpellValueAttempt++;
                        SetAreaCommonData("cutin_History",
SpellDataAttempt, SpellValueAttempt);
                        SaveCommonDataAreaA1("cutin_History");
                }



                let SpellBG = CreateSimple2DImageA2(0.79+0.01*max(0, 1-
num), SpellAttack_img, 0, 0, 255, 31);
```

```
                ObjSprite2D_SetDestRect(SpellBG, -224, -12, 32, 20);
                ObjRender_SetPosition(SpellBG, 340, 320+40*num, 0);
                ObjRender_SetAlpha(SpellBG, 0);
                ObjRender_SetColor(SpellBG, Colors[0][0], Colors[0][1],
Colors[0][2]);

                let SpellText = ObjText_Create;
                ObjText_SetText(SpellText,SpellName);
                ObjText_SetFontSize(SpellText,12);
                ObjText_SetFontColorTop(SpellText,255,255,255);
                ObjText_SetFontColorBottom(SpellText, Colors[0][0],
Colors[0][1], Colors[0][2]);
                ObjText_SetFontBorderType(SpellText,BORDER_FULL);
                ObjText_SetFontBorderColor(SpellText,32,32,32);
                ObjText_SetFontBorderWidth(SpellText,1);
                ObjText_SetHorizontalAlignment(SpellText,
ALIGNMENT_RIGHT);
                ObjText_SetMaxWidth(SpellText, GetStgFrameWidth-24);
                Obj_SetRenderPriority(SpellText, 0.79+0.01*max(0, 1-num));
                ObjRender_SetPosition(SpellText, 0, 320+40*num, 0);
                ObjRender_SetAlpha(SpellText, 0);


                let SpellInfo = ObjPrim_Create(OBJ_SPRITE_LIST_2D);
                let bonus = 0;
                let count = GetNumSize(bonus);
                let scount = max(2, GetNumSize(SpellValueAttempt));
                ObjPrim_SetTexture(SpellInfo, SpellAttack_img);
                ObjRender_SetBlendType(SpellInfo, BLEND_ALPHA);
                Obj_SetRenderPriority(SpellInfo, 0.8);
                ObjRender_SetY(SpellInfo, 36);
                ObjRender_SetAlpha(SpellInfo, 0);

                //-------------Introduction--------------------------

                loop(delay){yield;}
                let len = 30;
                ascent(i in 0..len){
                        let mod = i/len;
                        let tmod = min(1, ((i*2)/len));
                        ObjRender_SetScaleXYZ(SpellBG, 4-3*mod, 4-3*mod,
1);
                        ObjRender_SetAlpha(SpellBG, 255*mod);
                        ObjRender_SetX(SpellText, (GetStgFrameWidth-
24)*(1-mod));
```

```
                        //ObjRender_SetScaleXYZ(SpellText, 4-3*tmod, 4-
3*tmod, 1);
                        ObjRender_SetAlpha(SpellText, 255*tmod);
                        yield;
                }
                loop(40){yield;}
                let mov = 0;

                if(Type ==
"MOKOU"){SpellcardDeclareMovement([ObjRender_GetY(SpellText), 48+32*num,
0.175, 9]);loop(60){yield;}}
                else{RegularSpellCardDeclare([ObjRender_GetY(SpellText),
32+32*num, 0.135, 12]);}
                //32 is the offset from the top of the screen
                //If using a system where the lifebar is on the boss
rather than the top of the screen, 12 is a good value


                loop(delay*0.65){yield;}

                ascent(i in 0..80){
                        let endspeed = 0.5+i/2.5;
                        ObjRender_SetX(SpellBG,
ObjRender_GetX(SpellBG)+endspeed);
                        ObjRender_SetX(SpellText,
ObjRender_GetX(SpellText)+endspeed);
                        yield;
                }

                Obj_Delete(SpellBG);
                Obj_Delete(SpellText);

                function SpellcardDeclareMovement(dest){
                        ObjRender_SetY(SpellInfo, 24+dest[1]);
                        while(dest[0]>dest[1]){
                                dest[0] = ObjRender_GetY(SpellText);
                                mov = min(mov+dest[2], (dest[0]-
dest[1])/dest[3]+0.125);
                                ObjRender_SetY(SpellBG, dest[0]-mov);
                                ObjRender_SetY(SpellText, dest[0]-mov);
                                yield;
                        }
                }

                function RegularSpellCardDeclare(dest){
                        if(num!=0){Obj_Delete(SpellInfo);}
                        SpellcardDeclareMovement(dest);
                        ascent(i in 0..20){
                                ObjRender_SetAlpha(SpellInfo, 255/20*i);
                                DrawScoreBonus;
                                yield;
                        }
                        let timerf =
ObjEnemyBossScene_GetInfo(GetEnemyBossSceneObjectID, INFO_ORGTIMERF);
                        let startdiff = timerf-(timerf-5*60);
                        while(!Obj_IsDeleted(boss)){
                                ObjRender_SetAlpha(SpellBG, Alpha_HUD[0]);
                                ObjRender_SetAlpha(SpellText,
Alpha_HUD[0]);
                                ObjRender_SetAlpha(SpellInfo,
Alpha_HUD[0]);

                        //      let modif = min(1,
((ObjEnemyBossScene_GetInfo(GetEnemyBossSceneObjectID,
INFO_TIMERF))/(ObjEnemyBossScene_GetInfo(GetEnemyBossSceneObjectID,
INFO_ORGTIMERF)-startdiff)));
                        //      bonus =
((sscore*0.20)+(sscore*((modif)*0.80)));
                                DrawScoreBonus;
                                yield;
                        }
                        Obj_Delete(SpellInfo);


        if(bonus>=1&&ObjEnemyBossScene_GetInfo(GetEnemyBossSceneObjectID,
INFO_TIMERF)>0){
                        //      GotSpellCard(bonus);
                        //      AddScore(bonus);
                                if(!IsReplay&&!DEBUG&&num==0){
                                        SpellValueGet++;
                                        SetAreaCommonData("cutin_History",
SpellDataGet, SpellValueGet);

        SaveCommonDataAreaA1("cutin_History");
                                }
                        }else{
                        //      FailedSpellCard;
                        }
                }
                sub DrawScoreBonus{
                        if(GetPlayerY<128+32*length(spells)){Alpha_HUD[0]
= max(Alpha_HUD[0]-10, 85);}else{Alpha_HUD[0] = min(Alpha_HUD[0]+11,
255);}

                        ObjSpriteList2D_ClearVertexCount(SpellInfo);

                        ObjSpriteList2D_SetSourceRect(SpellInfo, 0, 48,
128, 60);
                        ObjSpriteList2D_SetDestCenter(SpellInfo);
                        ObjRender_SetX(SpellInfo, 382-130);
                        ObjSpriteList2D_AddVertex(SpellInfo);
                        //Bonus Score
                        bonus =
ObjEnemyBossScene_GetInfo(GetEnemyBossSceneObjectID, INFO_SPELL_SCORE);
                        count = GetNumSize(bonus);
                        let listNum = DigitToArray(min(99999999, bonus),
count);
                        let objScene = GetEnemyBossSceneObjectID;
                        if(ObjEnemyBossScene_GetInfo(objScene,
INFO_PLAYER_SHOOTDOWN_COUNT) +
                        ObjEnemyBossScene_GetInfo(objScene,
INFO_PLAYER_SPELL_COUNT) == 0){
                                ascent(iObj in 0 .. count){
                                        let num = listNum[iObj];
                                        ObjRender_SetX(SpellInfo, 382-94-
(8*count) + iObj * 7);

        ObjSpriteList2D_SetSourceRect(SpellInfo, num * 8, 64, (num + 1) *
8, 76);

        ObjSpriteList2D_SetDestCenter(SpellInfo);

        ObjSpriteList2D_AddVertex(SpellInfo);
                                }
                        }else{
                                        bonus = 0;
                                        ObjRender_SetX(SpellInfo, 382-131);

        ObjSpriteList2D_SetSourceRect(SpellInfo, 144, 48, 176, 60);

        ObjSpriteList2D_SetDestCenter(SpellInfo);

        ObjSpriteList2D_AddVertex(SpellInfo);
                        }
                        //History
                        listNum = DigitToArray(SpellValueGet, scount) ~
[10] ~ DigitToArray(SpellValueAttempt, scount);

                        ascent(iObj in 0 .. scount*2+1){
                                let num = listNum[iObj];
                                ObjRender_SetX(SpellInfo, 382-58 + iObj *
8);
                                ObjSpriteList2D_SetSourceRect(SpellInfo,
num * 8, 64, (num + 1) * 8, 76);
                                ObjSpriteList2D_SetDestCenter(SpellInfo);
                                ObjSpriteList2D_AddVertex(SpellInfo);
                        }
                }
        }

        task SpellAttackEffect(alphamax){
                if(IsSpellAttackAnimation){return;}
                IsSpellAttackAnimation = true;
                let alpha = 1;
                let GetCenterX = GetStgFrameWidth/2;
                let GetCenterY = GetStgFrameHeight/2;
                Octagon(GetCenterX*2, GetCenterY*2, -1.15, 180, 1);
                Octagon(GetCenterX*2, GetCenterY*2, 1.05, 240, 0);
                Octagon(GetCenterX*2, GetCenterY*2, 1.05, 295, 0);
                Octagon(GetCenterX/5, GetCenterY/5, 0.75, 170, 1);

                ascent (i in 0..5){
                        Line(GetCenterX, GetCenterY+i*75-75-95, 1.5);
                }
                ascent (i in 0..5){
                        Line(GetCenterX, GetCenterY+i*75-37.5-95, -1.5);
                }

                loop(15){yield;}
                ascent(i in 0..20){
                        alpha = 0.1+alphamax/20*i;
                        yield;
                }
                loop(110){yield;}
                descent(i in 0..20){
                        alpha = alphamax/20*i;
                        yield;
                }

                IsSpellAttackAnimation = false;
                task Line(mx, my, posinc){
                        let objSpellAttack = 
ObjPrim_Create(OBJ_SPRITE_2D);
                        ObjRender_SetBlendType(objSpellAttack, 
BLEND_ALPHA);
                        Obj_SetRenderPriority(objSpellAttack, 0.75);
                        ObjPrim_SetTexture(objSpellAttack, 
img_SpellAttack);
                        ObjSprite2D_SetSourceRect(objSpellAttack, 0, 0, 
512, 16);
                        ObjSprite2D_SetDestRect(objSpellAttack, -256, -8, 
256, 8);
                        let LineX = 0;
                        ObjRender_SetPosition(objSpellAttack, mx, my, 0);
                        ObjRender_SetAngleZ(objSpellAttack, -35);
                        while(alpha>0){
                                LineX += posinc;
                                ObjSprite2D_SetSourceRect(objSpellAttack, 
LineX, 0, 512+LineX, 16);
                                ObjRender_SetAlpha(objSpellAttack, alpha);
                                yield;
                        }
                        Obj_Delete(objSpellAttack);
                }

                task Octagon(mx, my, spininc, dist, size){

                        let countVertex = 16;
                        let listRadius = [];
                        loop(countVertex){
                                listRadius = listRadius ~ [0];
                        }

                        let objOutline = ObjPrim_Create(OBJ_PRIMITIVE_2D);
                        ObjPrim_SetPrimitiveType(objOutline, 
PRIMITIVE_TRIANGLESTRIP);
                        ObjPrim_SetVertexCount(objOutline, countVertex);
                        ObjRender_SetBlendType(objOutline, BLEND_ALPHA);
                        Obj_SetRenderPriority(objOutline, 0.76);
                        ObjPrim_SetTexture(objOutline, img_SpellAttack);

                        ascent (iVert in 0..countVertex/2){
                                let left = iVert * 128;
                                let indexVert = iVert * 2;
                                ObjPrim_SetVertexUVT(objOutline, indexVert 
+ 0, left, 0);
```

```
                                ObjPrim_SetVertexUVT(objOutline, indexVert
        + 1, left, 16);
                        }

                        let frame = 0;
                        let rRate = 1;
                        let spin = 0;

                        while(alpha>0){
                                spin+=spininc;
                                VertexSize;
                                ObjRender_SetPosition(objOutline, mx, my,
        0);

                                ObjRender_SetAngleZ(objOutline, spin);
                                ObjRender_SetAlpha(objOutline, alpha);
                                frame++;
                                yield;
                        }
                        Obj_Delete(objOutline);

                        task VertexSize{
                                if(frame>=35){return;}
                                ascent (iVert in 0..countVertex/2){
                                        let indexVert = iVert * 2;
                                        let angle = (360 / (countVertex / 2
        - 1) * iVert);

                                        let vx1 = listRadius[indexVert] *
        cos(angle);
                                        let vy1 = listRadius[indexVert] *
        sin(angle);

                ObjPrim_SetVertexPosition(objOutline, indexVert + 0, vx1, vy1, 0);

                                        let vx2 = listRadius[indexVert+1] *
        cos(angle);
                                        let vy2 = listRadius[indexVert+1] *
        sin(angle);

                ObjPrim_SetVertexPosition(objOutline, indexVert + 1, vx2, vy2, 0);

                                        let dr = (dist * rRate -
        listRadius[indexVert]) / 16;
                                        listRadius[indexVert] =
        listRadius[indexVert] + dr;
                                        if(frame>size){
```

```
                                                        listRadius[indexVert + 1] =
listRadius[indexVert + 1] + dr;
                                                }
                                        }
                                }
                        }
                }
        }

        task FadeInA1(obj, ftime){
                ascent(i in 0..ftime){
                        ObjRender_SetAlpha(obj, 255/ftime*i);
                        yield;
                }
        }
        task FadeInB1(obj, alpha, ftime){
                ascent(i in 0..ftime){
                        ObjRender_SetAlpha(obj, alpha/ftime*i);
                        yield;
                }
        }

        task FadeOutA1(obj, ftime){
                descent(i in 0..ftime){
                        ObjRender_SetAlpha(obj, 255/ftime*i);
                        yield;
                }
        }

        task FadeOutB1(obj, initalpha, ftime){
                descent(i in 0..ftime){
                        ObjRender_SetAlpha(obj, initalpha/ftime*i);
                        yield;
                }
        }



        function CreateSimple2DImageA1(pri, image){
                let obj = ObjPrim_Create(OBJ_SPRITE_2D);
                ObjRender_SetBlendType(obj, BLEND_ALPHA);
                Obj_SetRenderPriority(obj, pri);
                ObjPrim_SetTexture(obj, image);
                let test = [GetTextureWidth(image), GetTextureHeight(image)];
                ObjSprite2D_SetSourceRect(obj, 0, 0, test[0]-1, test[1]-1);
                ObjSprite2D_SetDestRect(obj, -(test[0]/2+0.5), -(test[1]/2),
        (test[0]/2), (test[1]/2));
                return obj;
        }

        function CreateSimple2DImageA2(pri, image, x1, y1, x2, y2){
                let obj = ObjPrim_Create(OBJ_SPRITE_2D);
                ObjRender_SetBlendType(obj, BLEND_ALPHA);
                Obj_SetRenderPriority(obj, pri);
                ObjPrim_SetTexture(obj, image);
                ObjSprite2D_SetSourceRect(obj, x1, y1, x2, y2);
                ObjSprite2D_SetDestCenter(obj);
                return obj;
        }


        function DigitToArray(digit, count){
                let res = [];
                digit = truncate(digit);

                loop{
                        let tnum = truncate(digit % 10);
                        digit /= 10;
                        res = [tnum] ~ res;
                        if(truncate(digit) == 0){break;}
                }

                loop(max(0, count - length(res))){
                        res = [0] ~ res;
                }

                return res;
        }

        function GetNumSize(value){
                if(value<=1){return 1;}
                else{return truncate(log10(value))+1;}
        }
```

GizmoSpriteLibrary.txt

```
task renderNueUFO(obj){

        let wi = 96;
        let he = 112;

        let dir;
        let speed;
        let whichway = 0;
        let frame = 0;
        let idleframe = 0;
        let castframe = 0;
        Obj_SetValue(obj,"cast",0);

        ObjPrim_SetTexture(obj,GetCurrentScriptDirectory ~
"./spriteimg/NueUFOSprite.png");
        ObjSprite2D_SetSourceRect(obj,0,0,wi,he);
        ObjSprite2D_SetDestCenter(obj);
        ObjRender_SetScaleXYZ(obj,1,1,0);

        while(!Obj_IsDeleted(obj)){

                dir = ObjMove_GetAngle(obj);
                speed = ObjMove_GetSpeed(obj);

                if(speed == 0 && frame > 0){whichway = -1;}
                else if(speed == 0 && frame < 0){whichway = 1;}
                else if(speed == 0 && frame == 0){whichway = 0;}

                else if(cos(dir) < 0){whichway=-1;}
                else if(cos(dir) > 0){whichway=1;}

                if(Obj_GetValueD(obj,"cast",0) == 0 && speed == 0 && frame
== 0){

                        castframe = 0;
                        if(idleframe >= 0 && idleframe <
10){ObjSprite2D_SetSourceRect(obj,0,0,wi,he);}
                        else if(idleframe >= 10 && idleframe <
20){ObjSprite2D_SetSourceRect(obj,wi,0,wi*2,he);}
                        else if(idleframe >= 20 && idleframe <
30){ObjSprite2D_SetSourceRect(obj,wi*2,0,wi*3,he);}
                        else if(idleframe >= 30 && idleframe <
40){ObjSprite2D_SetSourceRect(obj,wi*3,0,wi*4,he);}

                        if(idleframe < 40){idleframe++;}
                        else if(idleframe == 40){idleframe=0;}

                }

                else if(Obj_GetValueD(obj,"cast",0) == 1 && speed == 0 &&
frame == 0){
```

```
                        idleframe=0;
                        frame=0;
                        if(castframe >= 0 && castframe <
7){ObjSprite2D_SetSourceRect(obj,0,he*3,wi,he*4);}
                        if(castframe >= 7 && castframe <
14){ObjSprite2D_SetSourceRect(obj,wi,he*3,wi*2,he*4);}
                        if(castframe >= 14 && castframe <
21){ObjSprite2D_SetSourceRect(obj,wi*2,he*3,wi*3,he*4);}
                        if(castframe >= 21 && castframe <
23){ObjSprite2D_SetSourceRect(obj,wi*3,he*3,wi*4,he*4);}
                        if(castframe >= 23 && castframe <
25){ObjSprite2D_SetSourceRect(obj,wi*4,he*3,wi*5,he*4);}
                        if(castframe < 25){castframe++;}
                        else if(castframe == 25){castframe=21;}
                }

                else if(speed != 0 || frame != 0){

                        if(frame < 0 && frame >= -
8){ObjSprite2D_SetSourceRect(obj,0,he*2,wi,he*3);}
                        if(frame < -8 && frame >= -
16){ObjSprite2D_SetSourceRect(obj,wi,he*2,wi*2,he*3);}
                        if(frame < -16 && frame >= -
24){ObjSprite2D_SetSourceRect(obj,wi*2,he*2,wi*3,he*3);}
                        if(frame < -24 && frame >= -
25){ObjSprite2D_SetSourceRect(obj,wi*3,he*2,wi*4,he*3);}
                        if(frame > 0 && frame <=
8){ObjSprite2D_SetSourceRect(obj,0,he,wi,he*2);}
                        if(frame > 8 && frame <=
16){ObjSprite2D_SetSourceRect(obj,wi,he,wi*2,he*2);}
                        if(frame > 16 && frame <=
24){ObjSprite2D_SetSourceRect(obj,wi*2,he,wi*3,he*2);}
                        if(frame > 24 && frame <=
25){ObjSprite2D_SetSourceRect(obj,wi*3,he,wi*4,he*2);}
                        idleframe=0;
                        castframe=0;

                }

                frame=frame+whichway;
                if(frame > 25){frame=25;}
                if(frame < -25){frame=-25;}
                yield;

        }
```

```
}
```

KyunBullet.txt

```
#UserShotData

shot_image = "./img/KyunBullet.png"


//delay_rect = (0,574,32,606)  //starry orb
delay_rect = (0,606,32,638)  //star
//delay_rect = (32,574,64,606) //hollow star
//delay_rect = (32,606,64,637)  //orb

//ALPHA

// Bone
ShotData{ id=1101 rect=(0,0,20,30) render=ALPHA delay_color=
(128,128,128) } //Gray

//Artefakt
ShotData{ id=1102 render=ADD_ARGB fixed_angle=true delay_color=
(155,155,255)
        AnimationData{
                animation_data=(8,0,35,20,62)
                animation_data=(8,44,35,58,62)
                animation_data=(8,22,35,40,62)
        }
        collision = 8;
}

//Mamizou bird green
ShotData{ id=1103 render=ADD_ARGB fixed_angle=false delay_color=
(55,255,55)
        AnimationData{
                animation_data=(8,0,75,35,95)
                animation_data=(8,35,75,65,95)
                animation_data=(8,67,75,97,95)
        }
        collision = 8;
}

//Mamizou bird blue
ShotData{ id=1104 render=ADD_ARGB fixed_angle=false delay_color=
(55,55,255)
        AnimationData{
                animation_data=(8,0,95,35,117)
                animation_data=(8,35,95,65,117)
                animation_data=(8,67,95,97,117)
        }
        collision = 8;
}

//Mamizou bird yellow
ShotData{ id=1105 render=ADD_ARGB fixed_angle=false delay_color=
(155,155,55)
        AnimationData{
                animation_data=(8,0,117,35,139)
                animation_data=(8,35,117,65,139)
                animation_data=(8,67,117,97,139)
        }
        collision = 8;
}

//Mamizou bird red
ShotData{ id=1106 render=ADD_ARGB fixed_angle=false delay_color=
(255,155,55)
        AnimationData{
                animation_data=(8,0,140,35,160)
                animation_data=(8,35,140,65,160)
                animation_data=(8,67,140,97,160)
        }
        collision = 8;
}



//Mamizou bird B-green
ShotData{ id=1107 render=ADD_ARGB fixed_angle=false delay_color=
(55,255,55)
        AnimationData{
                animation_data=(8,103,75,134,95)
                animation_data=(8,134,75,166,95)
                animation_data=(8,167,75,197,95)
        }
        collision = 8;
}

//Mamizou bird B-blue
ShotData{ id=1108 render=ADD_ARGB fixed_angle=false delay_color=
(55,55,255)
        AnimationData{
                animation_data=(8,103,95,134,117)
                animation_data=(8,134,95,166,117)
                animation_data=(8,167,95,197,117)
        }
        collision = 8;
}

//Mamizou bird B-yellow
ShotData{ id=1109 render=ADD_ARGB fixed_angle=false delay_color=
(155,155,55))
        AnimationData{
                animation_data=(8,103,117,134,139)
                animation_data=(8,134,117,166,139)
                animation_data=(8,167,117,197,139)
        }
        collision = 8;
}


//Mamizou bird B-red
ShotData{ id=1110 render=ADD_ARGB fixed_angle=false delay_color=
(255,155,55)
        AnimationData{
                animation_data=(8,103,140,134,160)
                animation_data=(8,134,140,166,160)
                animation_data=(8,167,140,197,160)
        }
        collision = 8;
}

//Butterflys
ShotData{ id=1111 rect=(0,162,38,195) render=ALPHA delay_color=
(255,255,255) } // Black
ShotData{ id=1112 rect=(38,162,71,195) render=ALPHA delay_color=
(255,64,64) } // Red
ShotData{ id=1113 rect=(71,162,104,195) render=ALPHA delay_color=
(255,64,255) } // Purple
ShotData{ id=1114 rect=(104,162,137,195) render=ALPHA delay_color=
(64,64,255) } // Blue
ShotData{ id=1115 rect=(137,162,170,195) render=ALPHA delay_color=
(64,128,255) } // Aqua
ShotData{ id=1116 rect=(170,162,203,195) render=ALPHA delay_color=
(64,255,64) } // Green
ShotData{ id=1117 rect=(203,162,236,195) render=ALPHA delay_color=
(255,255,64) } // Yellow
ShotData{ id=1118 rect=(236,162,269,195) render=ALPHA delay_color=
(255,128,64) } // Orange
```

```
//Triangleblack
ShotData{ id=1119 rect=(12,198,29,218) render=ALPHA delay_color=
(255,255,255) } // Black
ShotData{ id=1120 rect=(29,198,46,218) render=ALPHA delay_color=
(255,64,64) } // Red
ShotData{ id=1121 rect=(46,198,62,218) render=ALPHA delay_color=
(255,64,255) } // Purple
ShotData{ id=1122 rect=(62,198,80,218) render=ALPHA delay_color=
(255,64,255) } // Pink
ShotData{ id=1123 rect=(80,198,96,218) render=ALPHA delay_color=
(64,64,255) } // Blue
ShotData{ id=1124 rect=(96,198,113,218) render=ALPHA delay_color=
(64,128,255) } // Aqua
ShotData{ id=1125 rect=(113,198,128,218) render=ALPHA delay_color=
(64,255,64) } // Green
ShotData{ id=1126 rect=(128,198,145,218) render=ALPHA delay_color=
(255,255,64) } // Yellow
ShotData{ id=1127 rect=(145,198,160,218) render=ALPHA delay_color=
(255,128,64) } // Orange
ShotData{ id=1128 rect=(160,198,177,218) render=ALPHA delay_color=
(0,0,0) } // White

//Trianglewhite
ShotData{ id=1129 rect=(12,218,29,236) render=ALPHA delay_color=
(255,255,255) } // Black
ShotData{ id=1130 rect=(29,218,46,236) render=ALPHA delay_color=
(255,64,64) } // Red
ShotData{ id=1131 rect=(46,218,62,236) render=ALPHA delay_color=
(255,64,255) } // Purple
ShotData{ id=1132 rect=(62,218,80,236) render=ALPHA delay_color=
(255,64,255) } // Pink
ShotData{ id=1133 rect=(80,218,96,236) render=ALPHA delay_color=
(64,64,255) } // Blue
ShotData{ id=1134 rect=(96,218,113,236) render=ALPHA delay_color=
(64,128,255) } // Aqua
ShotData{ id=1135 rect=(113,218,128,236) render=ALPHA delay_color=
(64,255,64) } // Green
ShotData{ id=1136 rect=(128,218,145,236) render=ALPHA delay_color=
(255,255,64) } // Yellow
ShotData{ id=1137 rect=(145,218,160,236) render=ALPHA delay_color=
(255,128,64) } // Orange
ShotData{ id=1138 rect=(160,218,177,236) render=ALPHA delay_color=
(0,0,0) } // White

//MagicCircle
```

```
ShotData{ id=1139 rect=(5,275,35,305) render=ALPHA angular_velocity = 5
delay_color= (255,255,255) } // Black
ShotData{ id=1140 rect=(35,275,64,305) render=ALPHA angular_velocity = 5
delay_color= (255,64,64) } // Red
ShotData{ id=1141 rect=(65,275,94,305) render=ALPHA angular_velocity = 5
delay_color= (255,64,255) } // Purple
ShotData{ id=1142 rect=(95,275,124,305) render=ALPHA angular_velocity = 5
delay_color= (64,64,255) } // Blue
ShotData{ id=1143 rect=(125,275,154,305) render=ALPHA angular_velocity =
5 delay_color= (64,128,255) } // Aqua
ShotData{ id=1144 rect=(154,275,183,305) render=ALPHA angular_velocity =
5 delay_color= (64,255,64) } // Green
ShotData{ id=1145 rect=(183,275,212,305) render=ALPHA angular_velocity =
5 delay_color= (255,255,64) } // Yellow
ShotData{ id=1146 rect=(212,275,241,305) render=ALPHA angular_velocity =
5 delay_color= (255,128,64) } // Orange

//Idostar

ShotData{
                id = 1147
                delay_color = (255,32,32)
                render = ALPHA
                collision = 1
                AnimationData{
                        animation_data = (3, 256, 256, 288, 288 )
                        animation_data = (3, 256, 288, 288, 320 )
                        animation_data = (3, 256, 320, 288, 352 )
                        animation_data = (3, 256, 352, 288, 384 )
                }
        }
        ShotData{
                id = 1148
                delay_color = (32,255,32)
                render = ALPHA
                collision = 1
                AnimationData{
                        animation_data = (3, 288, 256, 320, 288 )
                        animation_data = (3, 288, 288, 320, 320 )
                        animation_data = (3, 288, 320, 320, 352 )
                        animation_data = (3, 288, 352, 320, 384 )
                }
        }
        ShotData{
                id = 1149
                delay_color = (32,32,255)
                render = ALPHA
                collision = 1
                AnimationData{
                        animation_data = (3, 320, 256, 352, 288 )
                        animation_data = (3, 320, 288, 352, 320 )
                        animation_data = (3, 320, 320, 352, 352 )
                        animation_data = (3, 320, 352, 352, 384 )
                }
        }
        ShotData{
                id = 1150
                delay_color = (255,255,32)
                render = ALPHA
                collision = 1
                AnimationData{
                        animation_data = (3, 352, 256, 384, 288 )
                        animation_data = (3, 352, 288, 384, 320 )
                        animation_data = (3, 352, 320, 384, 352 )
                        animation_data = (3, 352, 352, 384, 384 )
                }
        }
        ShotData{
                id = 1151
                delay_color = (255,32,255)
                render = ALPHA
                collision = 1
                AnimationData{
                        animation_data = (3, 384, 256, 416, 288 )
                        animation_data = (3, 384, 288, 416, 320 )
                        animation_data = (3, 384, 320, 416, 352 )
                        animation_data = (3, 384, 352, 416, 384 )
                }
        }
        ShotData{
                id = 1152
                delay_color = (32,255,255)
                render = ALPHA
                collision = 1
                AnimationData{
                        animation_data = (3, 416, 256, 448, 288 )
                        animation_data = (3, 416, 288, 448, 320 )
                        animation_data = (3, 416, 320, 448, 352 )
                        animation_data = (3, 416, 352, 448, 384 )
                }
        }
        ShotData{
```

```
                id = 1153
                delay_color = (255,128,32)
                render = ALPHA
                collision = 1
                AnimationData{
                        animation_data = (3, 448, 256, 480, 288 )
                        animation_data = (3, 448, 288, 480, 320 )
                        animation_data = (3, 448, 320, 480, 352 )
                        animation_data = (3, 448, 352, 480, 384 )
                }
        }
        ShotData{
                id = 1154
                delay_color = (255,255,255)
                render = ALPHA
                collision = 1
                AnimationData{
                        animation_data = (3, 480, 256, 512, 288 )
                        animation_data = (3, 480, 288, 512, 320 )
                        animation_data = (3, 480, 320, 512, 352 )
                        animation_data = (3, 480, 352, 512, 384 )
                }
        }

//MagicCircle Neo
ShotData{ id=1155 rect=(5,315,35,345) render=ALPHA angular_velocity = 5
delay_color= (255,255,255) } // W
ShotData{ id=1156 rect=(35,315,64,345) render=ALPHA angular_velocity = 5
delay_color= (255,64,64) } // A
ShotData{ id=1157 rect=(65,315,94,345) render=ALPHA angular_velocity = 5
delay_color= (255,64,255) } // DGr
ShotData{ id=1158 rect=(95,315,124,345) render=ALPHA angular_velocity = 5
delay_color= (64,64,255) } // Y
ShotData{ id=1159 rect=(125,315,154,345) render=ALPHA angular_velocity =
5 delay_color= (64,128,255) } // R
ShotData{ id=1160 rect=(154,315,183,345) render=ALPHA angular_velocity =
5 delay_color= (64,255,64) } // P
ShotData{ id=1161 rect=(183,315,212,345) render=ALPHA angular_velocity =
5 delay_color= (255,255,64) } // DB
ShotData{ id=1162 rect=(212,315,241,345) render=ALPHA angular_velocity =
5 delay_color= (255,128,64) } // LB

/*//Nuclear
ShotData{ id=1163 rect=(440,4,471.5,35) render=ADD delay_color=
(255,64,64) } // Red
```

```
ShotData{ id=1164 rect=(471,4,503,35) render=ADD delay_color=
(255,128,64) } // Orange
ShotData{ id=1165 rect=(503,4,534,35) render=ADD delay_color=
(255,255,64) } // Yellow
ShotData{ id=1166 rect=(534,4,566,35) render=ADD delay_color= (64,255,64)
} // Green
ShotData{ id=1167 rect=(440,35,471.5,67) render=ADD delay_color=
(64,128,255) } // Aqua
ShotData{ id=1168 rect=(471,35,503,67) render=ADD delay_color=
(64,64,255) } // Blue
ShotData{ id=1169 rect=(503,35,534,67) render=ADD delay_color=
(255,64,255) } // Purple
ShotData{ id=1170 rect=(534,35,566,67) render=ADD delay_color= (0,0,0) }
// White*/

//MagicStar
ShotData{ id=1171 rect=(196,200,239,242) render=ALPHA angular_velocity =
5 delay_color= (255,255,255) } // W
ShotData{ id=1172 rect=(242,200,284,242) render=ALPHA angular_velocity =
5 delay_color= (255,64,64) } // A
ShotData{ id=1173 rect=(288,200,332,242) render=ALPHA angular_velocity =
5 delay_color= (255,64,255) } // DGr
ShotData{ id=1174 rect=(334,200,377,242) render=ALPHA angular_velocity =
5 delay_color= (64,64,255) } // Y
ShotData{ id=1175 rect=(380,200,422,242) render=ALPHA angular_velocity =
5 delay_color= (64,128,255) } // R
ShotData{ id=1176 rect=(436,200,468,242) render=ALPHA angular_velocity =
5 delay_color= (64,255,64) } // P
ShotData{ id=1177 rect=(472,200,514,242) render=ALPHA angular_velocity =
5 delay_color= (255,255,64) } // DB
ShotData{ id=1178 rect=(518,200,560,242) render=ALPHA angular_velocity =
5 delay_color= (255,128,64) } // LB

//Shio spear
ShotData{ id=1179 rect=(10,417,30,465) render=ALPHA delay_color=
(64,64,255) } // Blue

//Gaia Mallet
ShotData{ id=1180 rect=(45,395,80,462) render=ALPHA angular_velocity = 10
delay_color= (255,255,255) } // Blue
```

```
// A_Fumetsu

ShotData{ id=1181 rect=(358,4,378,28) render=ALPHA delay_color=
(255,255,255) } // Black
ShotData{ id=1182 rect=(378,4,401,28) render=ALPHA delay_color=
(255,64,64) } // Red
ShotData{ id=1183 rect=(401,4,424,28) render=ALPHA delay_color=
(255,64,255) } // Purple
ShotData{ id=1184 rect=(424,4,447,28) render=ALPHA delay_color=
(255,64,255) } // Magenta
ShotData{ id=1185 rect=(447,4,471,28) render=ALPHA delay_color=
(64,64,255) } // Blue
ShotData{ id=1186 rect=(471,4,495,28) render=ALPHA delay_color=
(64,128,255) } // Aqua
ShotData{ id=1187 rect=(495,4,517,28) render=ALPHA delay_color=
(64,255,64) } // Green
ShotData{ id=1188 rect=(517,4,542,28) render=ALPHA delay_color=
(255,255,64) } // Yellow
ShotData{ id=1189 rect=(542,4,566,28) render=ALPHA delay_color=
(255,128,64) } // Orange

// B_Fumetsu

ShotData{ id=1190 rect=(358,28,378,52) render=ALPHA delay_color=
(255,255,255) } // Black
ShotData{ id=1191 rect=(378,28,401,52) render=ALPHA delay_color=
(255,64,64) } // Red
ShotData{ id=1192 rect=(401,28,424,52) render=ALPHA delay_color=
(255,64,255) } // Purple
ShotData{ id=1193 rect=(424,28,447,52) render=ALPHA delay_color=
(255,64,255) } // Magenta
ShotData{ id=1194 rect=(447,28,471,52) render=ALPHA delay_color=
(64,64,255) } // Blue
ShotData{ id=1195 rect=(471,28,495,52) render=ALPHA delay_color=
(64,128,255) } // Aqua
ShotData{ id=1196 rect=(495,28,517,52) render=ALPHA delay_color=
(64,255,64) } // Green
ShotData{ id=1197 rect=(517,28,542,52) render=ALPHA delay_color=
(255,255,64) } // Yellow
ShotData{ id=1198 rect=(542,28,566,52) render=ALPHA delay_color=
(255,128,64) } // Orange

// C_Fumetsu
```

```
ShotData{ id=1199 rect=(358,52,378,76) render=ALPHA delay_color=
(255,255,255) } // Black
ShotData{ id=1200 rect=(378,52,401,76) render=ALPHA delay_color=
(255,64,64) } // Red
ShotData{ id=1201 rect=(401,52,424,76) render=ALPHA delay_color=
(255,64,255) } // Purple
ShotData{ id=1202 rect=(424,52,447,76) render=ALPHA delay_color=
(255,64,255) } // Magenta
ShotData{ id=1203 rect=(447,52,471,76) render=ALPHA delay_color=
(64,64,255) } // Blue
ShotData{ id=1204 rect=(471,52,495,76) render=ALPHA delay_color=
(64,128,255) } // Aqua
ShotData{ id=1205 rect=(495,52,517,76) render=ALPHA delay_color=
(64,255,64) } // Green
ShotData{ id=1206 rect=(517,52,542,76) render=ALPHA delay_color=
(255,255,64) } // Yellow
ShotData{ id=1207 rect=(542,52,566,76) render=ALPHA delay_color=
(255,128,64) } // Orange

// D_Fumetsu

ShotData{ id=1208 rect=(358,76,378,100) render=ALPHA delay_color=
(255,255,255) } // Black
ShotData{ id=1209 rect=(378,76,401,100) render=ALPHA delay_color=
(255,64,64) } // Red
ShotData{ id=1210 rect=(401,76,424,100) render=ALPHA delay_color=
(255,64,255) } // Purple
ShotData{ id=1211 rect=(424,76,447,100) render=ALPHA delay_color=
(255,64,255) } // Magenta
ShotData{ id=1212 rect=(447,76,471,100) render=ALPHA delay_color=
(64,64,255) } // Blue
ShotData{ id=1213 rect=(471,76,495,100) render=ALPHA delay_color=
(64,128,255) } // Aqua
ShotData{ id=1214 rect=(495,76,517,100) render=ALPHA delay_color=
(64,255,64) } // Green
ShotData{ id=1215 rect=(517,76,542,100) render=ALPHA delay_color=
(255,255,64) } // Yellow
ShotData{ id=1216 rect=(542,76,566,100) render=ALPHA delay_color=
(255,128,64) } // Orange

// ICESTAR

ShotData{ id=1217 rect=(88,408,139,458) render=ALPHA angular_velocity = 5
collision = 8 delay_color= (255,255,255) } // White
ShotData{ id=1218 rect=(139,408,192,458) render=ALPHA angular_velocity =
5 collision = 8 delay_color= (255,64,64) } // Red
```

```
ShotData{ id=1219 rect=(192,408,244,458) render=ALPHA angular_velocity =
5 collision = 8 delay_color= (255,64,255) } // Purple
ShotData{ id=1220 rect=(244,408,298,458) render=ALPHA angular_velocity =
5 collision = 8 delay_color= (64,64,255) } // Blue
ShotData{ id=1221 rect=(298,408,351,458) render=ALPHA angular_velocity =
5 collision = 8 delay_color= (64,128,255) } // Aqua
ShotData{ id=1222 rect=(351,408,405,458) render=ALPHA angular_velocity =
5 collision = 8 delay_color= (64,255,64) } // Green
ShotData{ id=1223 rect=(405,408,456,458) render=ALPHA angular_velocity =
5 collision = 8 delay_color= (255,255,64) } // Yellow
ShotData{ id=1224 rect=(456,408,508,458) render=ALPHA angular_velocity =
5 collision = 8 delay_color= (255,128,64) } // Orange

// KBL_SMALL

ShotData{ id=1225 rect=(393,104,411,122) render=ALPHA angular_velocity =
5 delay_color= (255,255,255) } // White
ShotData{ id=1226 rect=(410,104,429,122) render=ALPHA angular_velocity =
5 delay_color= (255,64,64) } // Red
ShotData{ id=1227 rect=(428,104,445,122) render=ALPHA angular_velocity =
5 delay_color= (255,64,255) } // Purple
ShotData{ id=1228 rect=(444,104,462,122) render=ALPHA angular_velocity =
5 delay_color= (64,64,255) } // Blue
ShotData{ id=1229 rect=(461,104,479,122) render=ALPHA angular_velocity =
5 delay_color= (64,128,255) } // Aqua
ShotData{ id=1230 rect=(478,104,496,122) render=ALPHA angular_velocity =
5 delay_color= (64,255,64) } // DGreen
ShotData{ id=1231 rect=(495,104,513,122) render=ALPHA angular_velocity =
5 delay_color= (255,255,64) } // Green
ShotData{ id=1232 rect=(512,104,530,122) render=ALPHA angular_velocity =
5 delay_color= (255,128,64) } // Yellow
ShotData{ id=1233 rect=(529,104,547,122) render=ALPHA angular_velocity =
5 delay_color= (255,128,64) } // Orange
ShotData{ id=1234 rect=(546,104,564,122) render=ALPHA angular_velocity =
5 delay_color= (255,128,64) } // Black

// CHUINOTO

//WHITE
ShotData{ id=1235 render=ALPHA fixed_angle=true delay_color=
(255,255,255)
        AnimationData{
                animation_data=(8,10,650,35,690)
                animation_data=(8,38,650,62,690)
                animation_data=(8,63,650,89,690)
                animation_data=(8,38,650,62,690)
        }
        collision = 8;
}

//RED
ShotData{ id=1236 render=ALPHA fixed_angle=true delay_color=
(255,255,255)
        AnimationData{
                animation_data=(8,10,690,35,725)
                animation_data=(8,35,690,62,725)
                animation_data=(8,62,690,89,725)
                animation_data=(8,35,690,62,725)
        }
        collision = 8;
}

//MAGENTA
ShotData{ id=1237 render=ALPHA fixed_angle=true delay_color=
(255,255,255)
        AnimationData{
                animation_data=(8,10,725,35,768)
                animation_data=(8,35,725,62,768)
                animation_data=(8,62,725,89,768)
                animation_data=(8,35,725,62,768)
        }
        collision = 8;
}

//PURPLE
ShotData{ id=1238 render=ALPHA fixed_angle=true delay_color=
(255,255,255)
        AnimationData{
                animation_data=(8,10,768,35,807)
                animation_data=(8,35,768,62,807)
                animation_data=(8,62,768,89,807)
                animation_data=(8,35,768,62,807)
        }
        collision = 8;
}

//BLUE
ShotData{ id=1239 render=ALPHA fixed_angle=true delay_color=
(255,255,255)
        AnimationData{
                animation_data=(8,10,807,35,850)
                animation_data=(8,35,807,62,850)
                animation_data=(8,62,807,89,850)
                animation_data=(8,35,807,62,850)
        }
        collision = 8;
}

//CYAN
ShotData{ id=1240 render=ALPHA fixed_angle=true delay_color=
(255,255,255)
        AnimationData{
                animation_data=(8,10,850,35,890)
                animation_data=(8,35,850,62,890)
                animation_data=(8,62,850,89,890)
                animation_data=(8,35,850,62,890)
        }
        collision = 8;
}

//AQUA
ShotData{ id=1241 render=ALPHA fixed_angle=true delay_color=
(255,255,255)
        AnimationData{
                animation_data=(8,10,890,35,931)
                animation_data=(8,35,890,62,931)
                animation_data=(8,62,890,89,931)
                animation_data=(8,35,890,62,931)
        }
        collision = 8;
}

//GREEN
ShotData{ id=1242 render=ALPHA fixed_angle=true delay_color=
(255,255,255)
        AnimationData{
                animation_data=(8,10,931,35,974)
                animation_data=(8,35,931,62,974)
                animation_data=(8,62,931,89,974)
                animation_data=(8,35,931,62,974)
        }
        collision = 8;
}

//YELLOW
ShotData{ id=1243 render=ALPHA fixed_angle=true delay_color=
(255,255,255)
        AnimationData{
                animation_data=(8,10,974,35,1015)
                animation_data=(8,35,974,62,1015)
                animation_data=(8,62,974,89,1015)
                animation_data=(8,35,974,62,1015)
        }
        collision = 8;
}

//ORANGE
ShotData{ id=1244 render=ALPHA fixed_angle=true delay_color=
(255,255,255)
        AnimationData{
                animation_data=(8,10,1015,35,1055)
                animation_data=(8,35,1015,62,1055)
                animation_data=(8,62,1015,89,1055)
                animation_data=(8,35,1015,62,1055)
        }
        collision = 8;
}

//BLACK
ShotData{ id=1245 render=ALPHA fixed_angle=true delay_color=
(255,255,255)
        AnimationData{
                animation_data=(8,10,1055,35,1095)
                animation_data=(8,35,1055,62,1095)
                animation_data=(8,62,1055,89,1095)
                animation_data=(8,35,1055,62,1095)
        }
        collision = 8;
}

ShotData{ id=1246 render=ALPHA fixed_angle=true delay_color=
(255,255,255) rect=(92,650,125,690) collision = 10 angular_velocity =
1}//WHITE
ShotData{ id=1247 render=ALPHA fixed_angle=true delay_color=
(255,255,255) rect=(92,690,125,725) collision = 10 angular_velocity = -
1}//RED
ShotData{ id=1248 render=ALPHA fixed_angle=true delay_color=
(255,255,255) rect=(92,725,125,768) collision = 10 angular_velocity =
1}//MAGENTA
ShotData{ id=1249 render=ALPHA fixed_angle=true delay_color=
(255,255,255) rect=(92,768,125,807) collision = 10 angular_velocity = -
1}//PURPLE
```

```
ShotData{ id=1250 render=ALPHA fixed_angle=true delay_color=
(255,255,255) rect=(92,807,125,850) collision = 10 angular_velocity =
1}//BLUE
ShotData{ id=1251 render=ALPHA fixed_angle=true delay_color=
(255,255,255) rect=(92,850,125,890) collision = 10 angular_velocity = -
1}//CYAN
ShotData{ id=1252 render=ALPHA fixed_angle=true delay_color=
(255,255,255) rect=(92,890,125,931) collision = 10 angular_velocity =
1}//AQUA
ShotData{ id=1253 render=ALPHA fixed_angle=true delay_color=
(255,255,255) rect=(92,931,125,974) collision = 10 angular_velocity = -
1}//GREEN
ShotData{ id=1254 render=ALPHA fixed_angle=true delay_color=
(255,255,255) rect=(92,974,125,1015) collision = 10 angular_velocity =
1}//YELLOW
ShotData{ id=1255 render=ALPHA fixed_angle=true delay_color=
(255,255,255) rect=(92,1015,125,1055) collision = 10 angular_velocity = -
1}//ORANGE
ShotData{ id=1256 render=ALPHA fixed_angle=true delay_color=
(255,255,255) rect=(92,1055,125,1095) collision = 10 angular_velocity =
1}//BLACK


//S_TEAR

ShotData{ id=1257 render=ALPHA delay_color= (255,255,255)
rect=(440,162,454,182)}//RED
ShotData{ id=1258 render=ALPHA delay_color= (255,255,255)
rect=(454,162,469,182)}//DRED
ShotData{ id=1259 render=ALPHA delay_color= (255,255,255)
rect=(469,162,484,182)}//MAGENTA
ShotData{ id=1260 render=ALPHA delay_color= (255,255,255)
rect=(484,162,499,182)}//BLUE
ShotData{ id=1261 render=ALPHA delay_color= (255,255,255)
rect=(500,162,515,182)}//AQUA
ShotData{ id=1262 render=ALPHA delay_color= (255,255,255)
rect=(516,162,530,182)}//GREEN
ShotData{ id=1263 render=ALPHA delay_color= (255,255,255)
rect=(530,162,545,182)}//YELLOW
ShotData{ id=1264 render=ALPHA delay_color= (255,255,255)
rect=(545,162,561,182)}//ORANGE

//TEAR

ShotData{ id=1265 render=ALPHA delay_color= (255,255,255)
rect=(370,128,395,157)}//RED
```

```
ShotData{ id=1266 render=ALPHA delay_color= (255,255,255)
rect=(395,128,418,157)}//DRED
ShotData{ id=1267 render=ALPHA delay_color= (255,255,255)
rect=(418,128,443,157)}//MAGENTA
ShotData{ id=1268 render=ALPHA delay_color= (255,255,255)
rect=(443,128,467,157)}//BLUE
ShotData{ id=1269 render=ALPHA delay_color= (255,255,255)
rect=(467,128,492,157)}//AQUA
ShotData{ id=1270 render=ALPHA delay_color= (255,255,255)
rect=(492,128,517,157)}//GREEN
ShotData{ id=1271 render=ALPHA delay_color= (255,255,255)
rect=(517,128,540,157)}//YELLOW
ShotData{ id=1272 render=ALPHA delay_color= (255,255,255)
rect=(540,128,564,157)}//ORANGE

//SPIKEHEAD

ShotData{ id=1273 render=ALPHA delay_color= (255,255,255)
rect=(6,244,28,270)}//LRED
ShotData{ id=1274 render=ALPHA delay_color= (255,255,255)
rect=(28,244,49,270)}//DRED
ShotData{ id=1275 render=ALPHA delay_color= (255,255,255)
rect=(49,244,70,270)}//MAGENTA
ShotData{ id=1276 render=ALPHA delay_color= (255,255,255)
rect=(70,244,91,270)}//PURPLE
ShotData{ id=1277 render=ALPHA delay_color= (255,255,255)
rect=(91,244,112,270)}//BLUE
ShotData{ id=1278 render=ALPHA delay_color= (255,255,255)
rect=(112,244,133,270)}//CYAN
ShotData{ id=1279 render=ALPHA delay_color= (255,255,255)
rect=(133,244,153,270)}//AQUA
ShotData{ id=1280 render=ALPHA delay_color= (255,255,255)
rect=(153,244,174,270)}//GREEN
ShotData{ id=1281 render=ALPHA delay_color= (255,255,255)
rect=(174,244,195,270)}//YELLOW
ShotData{ id=1282 render=ALPHA delay_color= (255,255,255)
rect=(195,244,216,270)}//ORANGE

//DANGO

ShotData{ id=1283 render=ALPHA delay_color= (255,255,255)
rect=(4,358,14,386)}//RED
ShotData{ id=1284 render=ALPHA delay_color= (255,255,255)
rect=(14,358,24,386)}//PURPLE
ShotData{ id=1285 render=ALPHA delay_color= (255,255,255)
rect=(24,358,34,386)}//BLUE
```

```
ShotData{ id=1286 render=ALPHA delay_color= (255,255,255)
rect=(34,358,44,386)}//AQUA
ShotData{ id=1287 render=ALPHA delay_color= (255,255,255)
rect=(44,358,54,386)}//GREEN
ShotData{ id=1288 render=ALPHA delay_color= (255,255,255)
rect=(54,358,64,386)}//YELLOW
ShotData{ id=1289 render=ALPHA delay_color= (255,255,255)
rect=(64,358,74,386)}//ORANGE

//LAVABALL

//RED
ShotData{ id=1290 render=ALPHA fixed_angle=false delay_color=
(255,255,255)
        AnimationData{
                animation_data=(5,150,650,194,712)
                animation_data=(5,194,650,235,712)
                animation_data=(5,235,650,277,712)
                animation_data=(5,194,650,235,712)
        }
        collision = 13;
}

//BLUE
ShotData{ id=1291 render=ALPHA fixed_angle=false delay_color=
(255,255,255)
        AnimationData{
                animation_data=(5,150,712,194,784)
                animation_data=(5,194,712,235,784)
                animation_data=(5,235,712,277,784)
                animation_data=(5,194,712,235,784)
        }
        collision = 13;
}

//AQUA
ShotData{ id=1292 render=ALPHA fixed_angle=false delay_color=
(255,255,255)
        AnimationData{
                animation_data=(5,150,784,194,852)
                animation_data=(5,194,784,235,852)
                animation_data=(5,235,784,277,852)
                animation_data=(5,194,784,235,852)
        }
        collision = 13;
}
```

```
//GREEN
ShotData{ id=1293 render=ALPHA fixed_angle=false delay_color=
(255,255,255)
        AnimationData{
                animation_data=(5,150,852,194,922)
                animation_data=(5,194,852,235,922)
                animation_data=(5,235,852,277,922)
                animation_data=(5,194,852,235,922)
        }
        collision = 13;
}

//YELLOW
ShotData{ id=1294 render=ALPHA fixed_angle=false delay_color=
(255,255,255)
        AnimationData{
                animation_data=(5,150,922,194,992)
                animation_data=(5,194,922,235,992)
                animation_data=(5,235,922,277,992)
                animation_data=(5,194,922,235,992)
        }
        collision = 13;
}

//S_ONMYOUGOKU

ShotData{ id=1295 render=ALPHA delay_color= (255,255,255)
angular_velocity=3 rect=(4,516,27,542)}//RED
ShotData{ id=1296 render=ALPHA delay_color= (255,255,255)
angular_velocity=-3  rect=(27,516,49,542)}//PURPLE
ShotData{ id=1297 render=ALPHA delay_color= (255,255,255)
angular_velocity=3  rect=(49,516,73,542)}//BLUE
ShotData{ id=1298 render=ALPHA delay_color= (255,255,255)
angular_velocity=-3  rect=(73,516,96,542)}//AQUA
ShotData{ id=1299 render=ALPHA delay_color= (255,255,255)
angular_velocity=3  rect=(96,516,118,542)}//GREEN
ShotData{ id=1300 render=ALPHA delay_color= (255,255,255)
angular_velocity=-3  rect=(118,516,142,542)}//YELLOW

//ONMYOUGOKU

ShotData{ id=1301 render=ALPHA delay_color= (255,255,255)
angular_velocity=-3 rect=(2,475,38,513)}//RED
ShotData{ id=1302 render=ALPHA delay_color= (255,255,255)
angular_velocity=3  rect=(38,475,78,513)}//PURPLE
```

```
ShotData{ id=1303 render=ALPHA delay_color= (255,255,255)
angular_velocity=-3  rect=(78,475,116,513)}//BLUE
ShotData{ id=1304 render=ALPHA delay_color= (255,255,255)
angular_velocity=3  rect=(116,475,153,513)}//AQUA
ShotData{ id=1305 render=ALPHA delay_color= (255,255,255)
angular_velocity=-3  rect=(153,475,192,513)}//GREEN
ShotData{ id=1306 render=ALPHA delay_color= (255,255,255)
angular_velocity=3  rect=(192,475,230,513)}//YELLOW

//SNOWFLAKE

ShotData{ id=1307 render=ALPHA delay_color= (255,255,255)
angular_velocity=3 rect=(4,544,29,572)}//WHITE
ShotData{ id=1308 render=ALPHA delay_color= (255,255,255)
angular_velocity=-3 rect=(29,544,54,572)}//RED
ShotData{ id=1309 render=ALPHA delay_color= (255,255,255)
angular_velocity=3 rect=(54,544,79,572)}//PURPLE
ShotData{ id=1310 render=ALPHA delay_color= (255,255,255)
angular_velocity=-3 rect=(79,544,104,572)}//BLUE
ShotData{ id=1311 render=ALPHA delay_color= (255,255,255)
angular_velocity=3 rect=(104,544,129,572)}//AQUA
ShotData{ id=1312 render=ALPHA delay_color= (255,255,255)
angular_velocity=-3 rect=(129,544,154,572)}//GREEN
ShotData{ id=1313 render=ALPHA delay_color= (255,255,255)
angular_velocity=3 rect=(154,544,179,572)}//YELLOW
ShotData{ id=1314 render=ALPHA delay_color= (255,255,255)
angular_velocity=-3 rect=(179,544,204,572)}//ORANGE

//SAKURA_BLOSSOM

ShotData{ id=1315 render=ALPHA delay_color= (255,255,255)
angular_velocity=4 rect=(73,575,92,602)}//WHITE
ShotData{ id=1316 render=ALPHA delay_color= (255,255,255)
angular_velocity=-4 rect=(92,575,110,602)}//RED
ShotData{ id=1317 render=ALPHA delay_color= (255,255,255)
angular_velocity=4 rect=(110,575,128,602)}//MAGENTA
ShotData{ id=1318 render=ALPHA delay_color= (255,255,255)
angular_velocity=-4 rect=(128,575,146,602)}//PURPLE
ShotData{ id=1319 render=ALPHA delay_color= (255,255,255)
angular_velocity=4 rect=(146,575,164,602)}//BLUE
ShotData{ id=1320 render=ALPHA delay_color= (255,255,255)
angular_velocity=-4 rect=(164,575,182,602)}//AQUA
ShotData{ id=1321 render=ALPHA delay_color= (255,255,255)
angular_velocity=4 rect=(182,575,200,602)}//GREEN
ShotData{ id=1322 render=ALPHA delay_color= (255,255,255)
angular_velocity=-4 rect=(200,575,218,602)}//YELLOW
```

```
ShotData{ id=1323 render=ALPHA delay_color= (255,255,255)
angular_velocity=4 rect=(218,575,236,602)}//ORANGE
ShotData{ id=1324 render=ALPHA delay_color= (255,255,255)
angular_velocity=-4 rect=(236,575,254,602)}//BLACK

//SAKURA_ARROW

ShotData{ id=1325 render=ALPHA delay_color= (255,255,255)
rect=(73,608,92,634)}//WHITE
ShotData{ id=1326 render=ALPHA delay_color= (255,255,255)
rect=(92,608,110,634)}//RED
ShotData{ id=1327 render=ALPHA delay_color= (255,255,255)
rect=(110,608,128,634)}//MAGENTA
ShotData{ id=1328 render=ALPHA delay_color= (255,255,255)
rect=(128,608,146,634)}//PURPLE
ShotData{ id=1329 render=ALPHA delay_color= (255,255,255)
rect=(146,608,164,634)}//BLUE
ShotData{ id=1330 render=ALPHA delay_color= (255,255,255)
rect=(164,608,182,634)}//AQUA
ShotData{ id=1331 render=ALPHA delay_color= (255,255,255)
rect=(182,608,200,634)}//GREEN
ShotData{ id=1332 render=ALPHA delay_color= (255,255,255)
rect=(200,608,218,634)}//YELLOW
ShotData{ id=1333 render=ALPHA delay_color= (255,255,255)
rect=(218,608,236,634)}//ORANGE
ShotData{ id=1334 render=ALPHA delay_color= (255,255,255)
rect=(236,608,254,634)}//BLACK


//ICEDIAMONDS

ShotData{ id=1335 render=ALPHA delay_color= (255,255,255)
AnimationData{
               animation_data=(10,70,40,86,70)
               animation_data=(10,70,10,86,40)
        }
}//WHITE

ShotData{ id=1336 render=ALPHA delay_color= (255,255,255)
AnimationData{
               animation_data=(10,86,40,102,70)
               animation_data=(10,86,10,102,40)
        }
}//LRED
```

```
ShotData{ id=1337 render=ALPHA delay_color= (255,255,255)
AnimationData{
		animation_data=(10,102,40,118,70)
		animation_data=(10,102,10,118,40)
	}
}//DRED


ShotData{ id=1338 render=ALPHA delay_color= (255,255,255)
AnimationData{
		animation_data=(10,118,40,135,70)
		animation_data=(10,118,10,135,40)
	}
}//MAGENTA


ShotData{ id=1339 render=ALPHA delay_color= (255,255,255)
AnimationData{
		animation_data=(10,135,40,150,70)
		animation_data=(10,135,10,150,40)
	}
}//PURPLE


ShotData{ id=1340 render=ALPHA delay_color= (255,255,255)
AnimationData{
		animation_data=(10,150,40,166,70)
		animation_data=(10,150,10,166,40)
	}
}//BLUE


ShotData{ id=1341 render=ALPHA delay_color= (255,255,255)
AnimationData{
		animation_data=(10,166,40,182,70)
		animation_data=(10,166,10,182,40)
	}
}//CYAN


ShotData{ id=1342 render=ALPHA delay_color= (255,255,255)
AnimationData{
		animation_data=(10,182,40,198,70)
		animation_data=(10,182,10,198,40)
	}
}//AQUA
```

```
ShotData{ id=1343 render=ALPHA delay_color= (255,255,255)
AnimationData{
               animation_data=(10,198,40,214,70)
               animation_data=(10,198,10,214,40)
       }
}//GREEN


ShotData{ id=1344 render=ALPHA delay_color= (255,255,255)
AnimationData{
               animation_data=(10,214,40,230,70)
               animation_data=(10,214,10,230,40)
       }
}//DGREEN


ShotData{ id=1345 render=ALPHA delay_color= (255,255,255)
AnimationData{
               animation_data=(10,230,40,246,70)
               animation_data=(10,230,10,246,40)
       }
}//YELLOW


ShotData{ id=1346 render=ALPHA delay_color= (255,255,255)
AnimationData{
               animation_data=(10,246,40,262,70)
               animation_data=(10,246,10,262,40)
       }
}//ORANGE


//STARBALL

ShotData{ id=1347 render=ALPHA delay_color= (255,255,255)
angular_velocity=7 rect=(270,168,294,192)}//RED
ShotData{ id=1348 render=ALPHA delay_color= (255,255,255)
angular_velocity=-7 rect=(294,168,318,192)}//PURPLE
ShotData{ id=1349 render=ALPHA delay_color= (255,255,255)
angular_velocity=7 rect=(318,168,342,192)}//BLUE
ShotData{ id=1350 render=ALPHA delay_color= (255,255,255)
angular_velocity=-7 rect=(342,168,366,192)}//AQUA
ShotData{ id=1351 render=ALPHA delay_color= (255,255,255)
angular_velocity=7 rect=(366,168,390,192)}//GREEN
```

```
ShotData{ id=1352 render=ALPHA delay_color= (255,255,255)
angular_velocity=-7 rect=(390,168,412,192)}//YELLOW
ShotData{ id=1353 render=ALPHA delay_color= (255,255,255)
angular_velocity=7 rect=(412,168,436,192)}//ORANGE


//STARRYORB

ShotData{ id=1354 render=ALPHA delay_color= (255,255,255)
angular_velocity=9 rect=(238,475,274,513)}//WHITE
ShotData{ id=1355 render=ALPHA delay_color= (255,255,255)
angular_velocity=9 rect=(274,475,310,513)}//RED
ShotData{ id=1356 render=ALPHA delay_color= (255,255,255)
angular_velocity=9 rect=(310,475,345,513)}//PURPLE
ShotData{ id=1357 render=ALPHA delay_color= (255,255,255)
angular_velocity=9 rect=(345,475,381,513)}//BLUE
ShotData{ id=1358 render=ALPHA delay_color= (255,255,255)
angular_velocity=9 rect=(381,475,418,513)}//AQUA
ShotData{ id=1359 render=ALPHA delay_color= (255,255,255)
angular_velocity=9 rect=(418,475,454,513)}//GREEN
ShotData{ id=1360 render=ALPHA delay_color= (255,255,255)
angular_velocity=9 rect=(454,475,490,513)}//YELLOW
ShotData{ id=1361 render=ALPHA delay_color= (255,255,255)
angular_velocity=9 rect=(490,475,525,513)}//ORANGE

ShotData{ id=1362 render=ALPHA delay_color= (255,255,255)
angular_velocity=-9 rect=(238,475,274,513)}//WHITE
ShotData{ id=1363 render=ALPHA delay_color= (255,255,255)
angular_velocity=-9 rect=(274,475,310,513)}//RED
ShotData{ id=1364 render=ALPHA delay_color= (255,255,255)
angular_velocity=-9 rect=(310,475,345,513)}//PURPLE
ShotData{ id=1365 render=ALPHA delay_color= (255,255,255)
angular_velocity=-9 rect=(345,475,381,513)}//BLUE
ShotData{ id=1366 render=ALPHA delay_color= (255,255,255)
angular_velocity=-9 rect=(381,475,418,513)}//AQUA
ShotData{ id=1367 render=ALPHA delay_color= (255,255,255)
angular_velocity=-9 rect=(418,475,454,513)}//GREEN
ShotData{ id=1368 render=ALPHA delay_color= (255,255,255)
angular_velocity=-9 rect=(454,475,490,513)}//YELLOW
ShotData{ id=1369 render=ALPHA delay_color= (255,255,255)
angular_velocity=-9 rect=(490,475,525,513)}//ORANGE

//YAJIRUSHI

ShotData{ id=1370 render=ALPHA delay_color= (255,255,255)
rect=(210,534,224,572)}//BLACK
```

```
ShotData{ id=1371 render=ALPHA delay_color= (255,255,255)
rect=(224,534,238,572)}//WHITE
ShotData{ id=1372 render=ALPHA delay_color= (255,255,255)
rect=(238,534,252,572)}//LRED
ShotData{ id=1373 render=ALPHA delay_color= (255,255,255)
rect=(252,534,265,572)}//RED
ShotData{ id=1374 render=ALPHA delay_color= (255,255,255)
rect=(265,534,278,572)}//LMAGENTA
ShotData{ id=1375 render=ALPHA delay_color= (255,255,255)
rect=(278,534,291,572)}//MAGENTA
ShotData{ id=1376 render=ALPHA delay_color= (255,255,255)
rect=(291,534,304,572)}//LPURPLE
ShotData{ id=1377 render=ALPHA delay_color= (255,255,255)
rect=(304,534,317,572)}//PURPLE
ShotData{ id=1378 render=ALPHA delay_color= (255,255,255)
rect=(317,534,330,572)}//LBLUE
ShotData{ id=1379 render=ALPHA delay_color= (255,255,255)
rect=(330,534,343,572)}//BLUE
ShotData{ id=1380 render=ALPHA delay_color= (255,255,255)
rect=(343,534,356,572)}//LAQUA
ShotData{ id=1381 render=ALPHA delay_color= (255,255,255)
rect=(356,534,369,572)}//AQUA
ShotData{ id=1382 render=ALPHA delay_color= (255,255,255)
rect=(369,534,382,572)}//LGREEN
ShotData{ id=1383 render=ALPHA delay_color= (255,255,255)
rect=(382,534,395,572)}//GREEN
ShotData{ id=1384 render=ALPHA delay_color= (255,255,255)
rect=(395,534,408,572)}//LYELLOW
ShotData{ id=1385 render=ALPHA delay_color= (255,255,255)
rect=(408,534,421,572)}//YELLOW
ShotData{ id=1386 render=ALPHA delay_color= (255,255,255)
rect=(421,534,434,572)}//LORANGE
ShotData{ id=1387 render=ALPHA delay_color= (255,255,255)
rect=(434,534,447,572)}//ORANGE

//KAMELOTCANDLE(S)

//KAMELOTCANDLE AA

ShotData{ id=1388 render=ALPHA delay_color= (255,255,255)
rect=(287,580,303,622) fixed_angle=true }//RED
ShotData{ id=1389 render=ALPHA delay_color= (255,255,255)
rect=(303,580,319,622) fixed_angle=true }//PURPLE
ShotData{ id=1390 render=ALPHA delay_color= (255,255,255)
rect=(319,580,335,622) fixed_angle=true }//BLUE
```

```
ShotData{ id=1391 render=ALPHA delay_color= (255,255,255)
rect=(335,580,351,622) fixed_angle=true }//AQUA
ShotData{ id=1392 render=ALPHA delay_color= (255,255,255)
rect=(351,580,367,622) fixed_angle=true }//MINT
ShotData{ id=1393 render=ALPHA delay_color= (255,255,255)
rect=(367,580,383,622) fixed_angle=true }//GREEN
ShotData{ id=1394 render=ALPHA delay_color= (255,255,255)
rect=(383,580,399,622) fixed_angle=true }//YELLOW
ShotData{ id=1395 render=ALPHA delay_color= (255,255,255)
rect=(399,580,415,622) fixed_angle=true }//ORANGE

//KAMELOTCANDLE AB

ShotData{ id=1396 render=ALPHA delay_color= (255,255,255)
rect=(287,580,303,622)}//RED
ShotData{ id=1397 render=ALPHA delay_color= (255,255,255)
rect=(303,580,319,622)}//PURPLE
ShotData{ id=1398 render=ALPHA delay_color= (255,255,255)
rect=(319,580,335,622)}//BLUE
ShotData{ id=1399 render=ALPHA delay_color= (255,255,255)
rect=(335,580,351,622)}//AQUA
ShotData{ id=1400 render=ALPHA delay_color= (255,255,255)
rect=(351,580,367,622)}//MINT
ShotData{ id=1401 render=ALPHA delay_color= (255,255,255)
rect=(367,580,383,622)}//GREEN
ShotData{ id=1402 render=ALPHA delay_color= (255,255,255)
rect=(383,580,399,622)}//YELLOW
ShotData{ id=1403 render=ALPHA delay_color= (255,255,255)
rect=(399,580,415,622)}//ORANGE

//KAMELOTCANDLE BA

ShotData{ id=1404 render=ALPHA delay_color= (255,255,255)
rect=(287,622,303,666) fixed_angle=true }//RED
ShotData{ id=1405 render=ALPHA delay_color= (255,255,255)
rect=(303,622,319,666) fixed_angle=true }//PURPLE
ShotData{ id=1406 render=ALPHA delay_color= (255,255,255)
rect=(319,622,335,666) fixed_angle=true }//BLUE
ShotData{ id=1407 render=ALPHA delay_color= (255,255,255)
rect=(335,622,351,666) fixed_angle=true }//AQUA
ShotData{ id=1408 render=ALPHA delay_color= (255,255,255)
rect=(351,622,367,666) fixed_angle=true }//MINT
ShotData{ id=1409 render=ALPHA delay_color= (255,255,255)
rect=(367,622,383,666) fixed_angle=true }//GREEN
ShotData{ id=1410 render=ALPHA delay_color= (255,255,255)
rect=(383,622,399,666) fixed_angle=true }//YELLOW
```

```
ShotData{ id=1411 render=ALPHA delay_color= (255,255,255)
rect=(399,622,415,666) fixed_angle=true }//ORANGE

//KAMELOTCANDLE BB

ShotData{ id=1412 render=ALPHA delay_color= (255,255,255)
rect=(287,622,303,666)}//RED
ShotData{ id=1413 render=ALPHA delay_color= (255,255,255)
rect=(303,622,319,666)}//PURPLE
ShotData{ id=1414 render=ALPHA delay_color= (255,255,255)
rect=(319,622,335,666)}//BLUE
ShotData{ id=1415 render=ALPHA delay_color= (255,255,255)
rect=(335,622,351,666)}//AQUA
ShotData{ id=1416 render=ALPHA delay_color= (255,255,255)
rect=(351,622,367,666)}//MINT
ShotData{ id=1417 render=ALPHA delay_color= (255,255,255)
rect=(367,622,383,666)}//GREEN
ShotData{ id=1418 render=ALPHA delay_color= (255,255,255)
rect=(383,622,399,666)}//YELLOW
ShotData{ id=1419 render=ALPHA delay_color= (255,255,255)
rect=(399,622,415,666)}//ORANGE

//KAMELOTCANDLE CA

ShotData{ id=1420 render=ALPHA delay_color= (255,255,255)
rect=(287,666,303,710) fixed_angle=true }//RED
ShotData{ id=1421 render=ALPHA delay_color= (255,255,255)
rect=(303,666,319,710) fixed_angle=true }//PURPLE
ShotData{ id=1422 render=ALPHA delay_color= (255,255,255)
rect=(319,666,335,710) fixed_angle=true }//BLUE
ShotData{ id=1423 render=ALPHA delay_color= (255,255,255)
rect=(335,666,351,710) fixed_angle=true }//AQUA
ShotData{ id=1424 render=ALPHA delay_color= (255,255,255)
rect=(351,666,367,710) fixed_angle=true }//MINT
ShotData{ id=1425 render=ALPHA delay_color= (255,255,255)
rect=(367,666,383,710) fixed_angle=true }//GREEN
ShotData{ id=1426 render=ALPHA delay_color= (255,255,255)
rect=(383,666,399,710) fixed_angle=true }//YELLOW
ShotData{ id=1427 render=ALPHA delay_color= (255,255,255)
rect=(399,666,415,710) fixed_angle=true }//ORANGE

//KAMELOTCANDLE CB

ShotData{ id=1428 render=ALPHA delay_color= (255,255,255)
rect=(287,666,303,710)}//RED
```

```
ShotData{ id=1429 render=ALPHA delay_color= (255,255,255)
rect=(303,666,319,710)}//PURPLE
ShotData{ id=1430 render=ALPHA delay_color= (255,255,255)
rect=(319,666,335,710)}//BLUE
ShotData{ id=1431 render=ALPHA delay_color= (255,255,255)
rect=(335,666,351,710)}//AQUA
ShotData{ id=1432 render=ALPHA delay_color= (255,255,255)
rect=(351,666,367,710)}//MINT
ShotData{ id=1433 render=ALPHA delay_color= (255,255,255)
rect=(367,666,383,710)}//GREEN
ShotData{ id=1434 render=ALPHA delay_color= (255,255,255)
rect=(383,666,399,710)}//YELLOW
ShotData{ id=1435 render=ALPHA delay_color= (255,255,255)
rect=(399,666,415,710)}//ORANGE

//KAMELOTCANDLE DA

ShotData{ id=1436 render=ALPHA delay_color= (255,255,255)
rect=(287,710,303,755) fixed_angle=true }//RED
ShotData{ id=1437 render=ALPHA delay_color= (255,255,255)
rect=(303,710,319,755) fixed_angle=true }//PURPLE
ShotData{ id=1438 render=ALPHA delay_color= (255,255,255)
rect=(319,710,335,755) fixed_angle=true }//BLUE
ShotData{ id=1439 render=ALPHA delay_color= (255,255,255)
rect=(335,710,351,755) fixed_angle=true }//AQUA
ShotData{ id=1440 render=ALPHA delay_color= (255,255,255)
rect=(351,710,367,755) fixed_angle=true }//MINT
ShotData{ id=1441 render=ALPHA delay_color= (255,255,255)
rect=(367,710,383,755) fixed_angle=true }//GREEN
ShotData{ id=1442 render=ALPHA delay_color= (255,255,255)
rect=(383,710,399,755) fixed_angle=true }//YELLOW
ShotData{ id=1443 render=ALPHA delay_color= (255,255,255)
rect=(399,710,415,755) fixed_angle=true }//ORANGE

//KAMELOTCANDLE DB

ShotData{ id=1444 render=ALPHA delay_color= (255,255,255)
rect=(287,710,303,755)}//RED
ShotData{ id=1445 render=ALPHA delay_color= (255,255,255)
rect=(303,710,319,755)}//PURPLE
ShotData{ id=1446 render=ALPHA delay_color= (255,255,255)
rect=(319,710,335,755)}//BLUE
ShotData{ id=1447 render=ALPHA delay_color= (255,255,255)
rect=(335,710,351,755)}//AQUA
ShotData{ id=1448 render=ALPHA delay_color= (255,255,255)
rect=(351,710,367,755)}//MINT
```

```
ShotData{ id=1449 render=ALPHA delay_color= (255,255,255)
rect=(367,710,383,755)}//GREEN
ShotData{ id=1450 render=ALPHA delay_color= (255,255,255)
rect=(383,710,399,755)}//YELLOW
ShotData{ id=1451 render=ALPHA delay_color= (255,255,255)
rect=(399,710,415,755)}//ORANGE

//BLUBBLE

ShotData{ id=1452 render=ALPHA delay_color= (255,255,255)
rect=(468,520,540,592) angular_velocity=2}//WHITE
ShotData{ id=1453 render=ALPHA delay_color= (255,255,255)
rect=(468,592,540,666) angular_velocity=-2}//RED
ShotData{ id=1454 render=ALPHA delay_color= (255,255,255)
rect=(468,666,540,738) angular_velocity=2}//PURPLE
ShotData{ id=1455 render=ALPHA delay_color= (255,255,255)
rect=(468,738,540,812) angular_velocity=-2}//BLUE
ShotData{ id=1456 render=ALPHA delay_color= (255,255,255)
rect=(468,812,540,884) angular_velocity=2}//AQUA
ShotData{ id=1457 render=ALPHA delay_color= (255,255,255)
rect=(468,884,540,956) angular_velocity=-2}//GREEN
ShotData{ id=1458 render=ALPHA delay_color= (255,255,255)
rect=(468,956,540,1028) angular_velocity=2}//YELLOW
ShotData{ id=1459 render=ALPHA delay_color= (255,255,255)
rect=(468,1028,540,1100) angular_velocity=-2}//ORANGE

//S_BLUBBLE

ShotData{ id=1460 render=ALPHA delay_color= (255,255,255)
rect=(428,586,454,613) angular_velocity=-1}//WHITE
ShotData{ id=1461 render=ALPHA delay_color= (255,255,255)
rect=(428,613,454,641) angular_velocity=1}//RED
ShotData{ id=1462 render=ALPHA delay_color= (255,255,255)
rect=(428,641,454,669) angular_velocity=-1}//PURPLE
ShotData{ id=1463 render=ALPHA delay_color= (255,255,255)
rect=(428,669,454,697) angular_velocity=1}//BLUE
ShotData{ id=1464 render=ALPHA delay_color= (255,255,255)
rect=(428,697,454,724) angular_velocity=-1}//AQUA
ShotData{ id=1465 render=ALPHA delay_color= (255,255,255)
rect=(428,724,454,751) angular_velocity=1}//GREEN
ShotData{ id=1466 render=ALPHA delay_color= (255,255,255)
rect=(428,751,454,779) angular_velocity=-1}//YELLOW
ShotData{ id=1467 render=ALPHA delay_color= (255,255,255)
rect=(428,779,454,806) angular_velocity=1}//ORANGE

//STICK
```

```
//RED
ShotData{ id=1468 render=ALPHA delay_color= (255,255,255)
AnimationData{
                animation_data=(10,290,758,303,787)
                animation_data=(10,290,787,303,816)
        }
}

//MAGENTA
ShotData{ id=1469 render=ALPHA delay_color= (255,255,255)
AnimationData{
                animation_data=(10,303,758,316,787)
                animation_data=(10,303,787,316,816)
        }
}

//PURPLE
ShotData{ id=1470 render=ALPHA delay_color= (255,255,255)
AnimationData{
                animation_data=(10,316,758,329,787)
                animation_data=(10,316,787,329,816)
        }
}

//BLUE
ShotData{ id=1471 render=ALPHA delay_color= (255,255,255)
AnimationData{
                animation_data=(10,329,758,342,787)
                animation_data=(10,329,787,342,816)
        }
}

//AQUA
ShotData{ id=1472 render=ALPHA delay_color= (255,255,255)
AnimationData{
                animation_data=(10,342,758,355,787)
                animation_data=(10,342,787,355,816)
        }
}

//MINT
ShotData{ id=1473 render=ALPHA delay_color= (255,255,255)
AnimationData{
                animation_data=(10,355,758,368,787)
                animation_data=(10,355,787,368,816)
        }
}

//GREEN
ShotData{ id=1474 render=ALPHA delay_color= (255,255,255)
AnimationData{
                animation_data=(10,368,758,381,787)
                animation_data=(10,368,787,381,816)
        }
}

//YELLOW
ShotData{ id=1475 render=ALPHA delay_color= (255,255,255)
AnimationData{
                animation_data=(10,381,758,394,787)
                animation_data=(10,381,787,394,816)
        }
}

//ORANGE
ShotData{ id=1476 render=ALPHA delay_color= (255,255,255)
AnimationData{
                animation_data=(10,394,758,407,787)
                animation_data=(10,394,787,407,816)
        }
}


//KOKORO

ShotData{ id=1477 render=ALPHA delay_color= (255,255,255)
rect=(206,72,236,102)}//RED
ShotData{ id=1478 render=ALPHA delay_color= (255,255,255)
rect=(236,72,268,102)}//DRED
ShotData{ id=1479 render=ALPHA delay_color= (255,255,255)
rect=(206,102,236,130)}//MAGENTA
ShotData{ id=1480 render=ALPHA delay_color= (255,255,255)
rect=(236,102,268,130)}//DMAGENTA
ShotData{ id=1481 render=ALPHA delay_color= (255,255,255)
rect=(206,130,236,160)}//BLUE
ShotData{ id=1482 render=ALPHA delay_color= (255,255,255)
rect=(236,130,268,160)}//DBLUE
ShotData{ id=1483 render=ALPHA delay_color= (255,255,255)
rect=(268,72,299,102)}//AQUA
ShotData{ id=1484 render=ALPHA delay_color= (255,255,255)
rect=(299,72,330,102)}//DAQUA
```

```
ShotData{ id=1485 render=ALPHA delay_color= (255,255,255)
rect=(268,102,299,130)}//GREEN
ShotData{ id=1486 render=ALPHA delay_color= (255,255,255)
rect=(299,102,330,130)}//DGREEN
ShotData{ id=1487 render=ALPHA delay_color= (255,255,255)
rect=(268,130,299,160)}//YELLOW
ShotData{ id=1488 render=ALPHA delay_color= (255,255,255)
rect=(299,130,330,160)}//DYELLOW

//FLOWER

ShotData{ id=1489 render=ALPHA delay_color= (255,255,255)
rect=(286,818,314,848) angular_velocity=-5}//RED
ShotData{ id=1490 render=ALPHA delay_color= (255,255,255)
rect=(314,818,342,848) angular_velocity=5}//PURPLE
ShotData{ id=1491 render=ALPHA delay_color= (255,255,255)
rect=(342,818,370,848) angular_velocity=-5}//BLUE
ShotData{ id=1492 render=ALPHA delay_color= (255,255,255)
rect=(370,818,398,848) angular_velocity=5}//AQUA
ShotData{ id=1493 render=ALPHA delay_color= (255,255,255)
rect=(398,818,426,848) angular_velocity=-5}//GREEN
ShotData{ id=1494 render=ALPHA delay_color= (255,255,255)
rect=(426,818,456,848) angular_velocity=5}//YELLOW

//LEAF

ShotData{ id=1495 render=ALPHA delay_color= (255,255,255)
rect=(144,996,172,1028) angular_velocity=2}//WHITE
ShotData{ id=1496 render=ALPHA delay_color= (255,255,255)
rect=(172,996,199,1028) angular_velocity=-2}//RED
ShotData{ id=1497 render=ALPHA delay_color= (255,255,255)
rect=(199,996,226,1028) angular_velocity=2}//PURPLE
ShotData{ id=1498 render=ALPHA delay_color= (255,255,255)
rect=(226,996,253,1028) angular_velocity=-2}//BLUE
ShotData{ id=1499 render=ALPHA delay_color= (255,255,255)
rect=(253,996,282,1028) angular_velocity=2}//AQUA
ShotData{ id=1500 render=ALPHA delay_color= (255,255,255)
rect=(282,996,309,1028) angular_velocity=-2}//GREEN
ShotData{ id=1501 render=ALPHA delay_color= (255,255,255)
rect=(309,996,336,1028) angular_velocity=2}//YELLOW
ShotData{ id=1502 render=ALPHA delay_color= (255,255,255)
rect=(336,996,364,1028) angular_velocity=-2}//ORANGE

//KNIFES
```

```
ShotData{ id=1519 render=ALPHA delay_color= (255,255,255)
rect=(590,80,616,130)}//WHITE
ShotData{ id=1520 render=ALPHA delay_color= (255,255,255)
rect=(616,80,642,130)}//RED
ShotData{ id=1521 render=ALPHA delay_color= (255,255,255)
rect=(642,80,668,130)}//PURPLE
ShotData{ id=1522 render=ALPHA delay_color= (255,255,255)
rect=(668,80,694,130)}//BLUE
ShotData{ id=1523 render=ALPHA delay_color= (255,255,255)
rect=(694,80,720,130)}//CYAN
ShotData{ id=1524 render=ALPHA delay_color= (255,255,255)
rect=(720,80,744,130)}//AQUA
ShotData{ id=1525 render=ALPHA delay_color= (255,255,255)
rect=(744,80,770,130)}//GREEN
ShotData{ id=1526 render=ALPHA delay_color= (255,255,255)
rect=(770,80,796,130)}//YELLOW
ShotData{ id=1527 render=ALPHA delay_color= (255,255,255)
rect=(796,80,820,130)}//ORANGE

//C_KNIFES

ShotData{ id=1528 render=ALPHA delay_color= (255,255,255)
rect=(590,130,616,180)}//WHITE
ShotData{ id=1529 render=ALPHA delay_color= (255,255,255)
rect=(616,130,642,180)}//RED
ShotData{ id=1530 render=ALPHA delay_color= (255,255,255)
rect=(642,130,668,180)}//PURPLE
ShotData{ id=1531 render=ALPHA delay_color= (255,255,255)
rect=(668,130,694,180)}//BLUE
ShotData{ id=1532 render=ALPHA delay_color= (255,255,255)
rect=(694,130,720,180)}//CYAN
ShotData{ id=1533 render=ALPHA delay_color= (255,255,255)
rect=(720,130,744,180)}//AQUA
ShotData{ id=1534 render=ALPHA delay_color= (255,255,255)
rect=(744,130,770,180)}//GREEN
ShotData{ id=1535 render=ALPHA delay_color= (255,255,255)
rect=(770,130,796,180)}//YELLOW
ShotData{ id=1536 render=ALPHA delay_color= (255,255,255)
rect=(796,130,820,180)}//ORANGE

//SWORD

ShotData{ id=1537 render=ALPHA delay_color= (255,255,255)
rect=(590,180,616,266) collision=13}//WHITE
ShotData{ id=1538 render=ALPHA delay_color= (255,255,255)
rect=(616,180,642,266) collision=13}//RED
```

```
ShotData{ id=1539 render=ALPHA delay_color= (255,255,255)
rect=(642,180,668,266) collision=13}//PURPLE
ShotData{ id=1540 render=ALPHA delay_color= (255,255,255)
rect=(668,180,694,266) collision=13}//BLUE
ShotData{ id=1541 render=ALPHA delay_color= (255,255,255)
rect=(694,180,720,266) collision=13}//CYAN
ShotData{ id=1542 render=ALPHA delay_color= (255,255,255)
rect=(720,180,744,266) collision=13}//AQUA
ShotData{ id=1543 render=ALPHA delay_color= (255,255,255)
rect=(744,180,770,266) collision=13}//GREEN
ShotData{ id=1544 render=ALPHA delay_color= (255,255,255)
rect=(770,180,796,266) collision=13}//YELLOW
ShotData{ id=1545 render=ALPHA delay_color= (255,255,255)
rect=(796,180,820,266) collision=13}//ORANGE

//C_SWORD

ShotData{ id=1546 render=ALPHA delay_color= (255,255,255)
rect=(590,266,616,344) collision=13}//WHITE
ShotData{ id=1547 render=ALPHA delay_color= (255,255,255)
rect=(616,266,642,344) collision=13}//RED
ShotData{ id=1548 render=ALPHA delay_color= (255,255,255)
rect=(642,266,668,344) collision=13}//PURPLE
ShotData{ id=1549 render=ALPHA delay_color= (255,255,255)
rect=(668,266,694,344) collision=13}//BLUE
ShotData{ id=1550 render=ALPHA delay_color= (255,255,255)
rect=(694,266,720,344) collision=13}//CYAN
ShotData{ id=1551 render=ALPHA delay_color= (255,255,255)
rect=(720,266,744,344) collision=13}//AQUA
ShotData{ id=1552 render=ALPHA delay_color= (255,255,255)
rect=(744,266,770,344) collision=13}//GREEN
ShotData{ id=1553 render=ALPHA delay_color= (255,255,255)
rect=(770,266,796,344) collision=13}//YELLOW
ShotData{ id=1554 render=ALPHA delay_color= (255,255,255)
rect=(796,266,820,344) collision=13}//ORANGE

//SHURIKEN

ShotData{ id=1555 render=ALPHA delay_color= (255,255,255)
rect=(588,348,646,412) angular_velocity=-9}//BLACK
ShotData{ id=1556 render=ALPHA delay_color= (255,255,255)
rect=(646,348,702,412) angular_velocity=9}//RED
ShotData{ id=1557 render=ALPHA delay_color= (255,255,255)
rect=(702,348,760,412) angular_velocity=-9}//PURPLE
ShotData{ id=1558 render=ALPHA delay_color= (255,255,255)
rect=(760,348,816,412) angular_velocity=9}//BLUE
```

```
ShotData{ id=1559 render=ALPHA delay_color= (255,255,255)
rect=(588,412,646,478) angular_velocity=-9}//AQUA
ShotData{ id=1560 render=ALPHA delay_color= (255,255,255)
rect=(646,412,702,478) angular_velocity=9}//GREEN
ShotData{ id=1561 render=ALPHA delay_color= (255,255,255)
rect=(702,412,760,478) angular_velocity=-9}//YELLOW
ShotData{ id=1562 render=ALPHA delay_color= (255,255,255)
rect=(760,412,816,478) angular_velocity=9}//ORANGE

//S_SHURIKEN

ShotData{ id=1563 render=ALPHA delay_color= (255,255,255)
rect=(602,486,627,516) angular_velocity=-9}//BLACK
ShotData{ id=1564 render=ALPHA delay_color= (255,255,255)
rect=(627,486,652,516) angular_velocity=9}//RED
ShotData{ id=1565 render=ALPHA delay_color= (255,255,255)
rect=(652,486,677,516) angular_velocity=-9}//PURPLE
ShotData{ id=1566 render=ALPHA delay_color= (255,255,255)
rect=(677,486,702,516) angular_velocity=9}//BLUE
ShotData{ id=1567 render=ALPHA delay_color= (255,255,255)
rect=(702,486,726,516) angular_velocity=-9}//AQUA
ShotData{ id=1568 render=ALPHA delay_color= (255,255,255)
rect=(726,486,751,516) angular_velocity=9}//GREEN
ShotData{ id=1569 render=ALPHA delay_color= (255,255,255)
rect=(751,486,776,516) angular_velocity=-9}//YELLOW
ShotData{ id=1570 render=ALPHA delay_color= (255,255,255)
rect=(776,486,802,516) angular_velocity=9}//ORANGE

//S_PENTAGON

ShotData{ id=1571 render=ALPHA delay_color= (255,255,255)
rect=(89,373,115,399) angular_velocity=4}//RED
ShotData{ id=1572 render=ALPHA delay_color= (255,255,255)
rect=(118,373,144,399) angular_velocity=-4}//PURPLE
ShotData{ id=1573 render=ALPHA delay_color= (255,255,255)
rect=(147,373,173,399) angular_velocity=4}//BLUE
ShotData{ id=1574 render=ALPHA delay_color= (255,255,255)
rect=(176,373,202,399) angular_velocity=-4}//GREEN
ShotData{ id=1575 render=ALPHA delay_color= (255,255,255)
rect=(206,373,231,399) angular_velocity=4}//YELLOW

//GEAR

ShotData{ id=1576 render=ALPHA delay_color= (255,255,255)
rect=(600,523,649,572) angular_velocity=4}//RED
```

```
ShotData{ id=1577 render=ALPHA delay_color= (255,255,255)
rect=(649,523,697,572) angular_velocity=4}//PURPLE
ShotData{ id=1578 render=ALPHA delay_color= (255,255,255)
rect=(697,523,747,572) angular_velocity=4}//BLUE
ShotData{ id=1579 render=ALPHA delay_color= (255,255,255)
rect=(747,523,795,572) angular_velocity=4}//AQUA
ShotData{ id=1580 render=ALPHA delay_color= (255,255,255)
rect=(600,572,649,621) angular_velocity=4}//GREEN
ShotData{ id=1581 render=ALPHA delay_color= (255,255,255)
rect=(649,572,697,621) angular_velocity=4}//YELLOW
ShotData{ id=1582 render=ALPHA delay_color= (255,255,255)
rect=(697,572,747,621) angular_velocity=4}//ORANGE
ShotData{ id=1583 render=ALPHA delay_color= (255,255,255)
rect=(747,572,795,621) angular_velocity=4}//WHITE

//S_GEAR

ShotData{ id=1584 render=ALPHA delay_color= (255,255,255)
rect=(602,634,625,657) angular_velocity=4}//RED
ShotData{ id=1585 render=ALPHA delay_color= (255,255,255)
rect=(626,634,649,657) angular_velocity=4}//PURPLE
ShotData{ id=1586 render=ALPHA delay_color= (255,255,255)
rect=(650,634,674,657) angular_velocity=4}//BLUE
ShotData{ id=1587 render=ALPHA delay_color= (255,255,255)
rect=(675,634,699,657) angular_velocity=4}//AQUA
ShotData{ id=1588 render=ALPHA delay_color= (255,255,255)
rect=(701,634,724,657) angular_velocity=4}//GREEN
ShotData{ id=1589 render=ALPHA delay_color= (255,255,255)
rect=(725,634,748,657) angular_velocity=4}//YELLOW
ShotData{ id=1590 render=ALPHA delay_color= (255,255,255)
rect=(750,634,772,657) angular_velocity=4}//ORANGE
ShotData{ id=1591 render=ALPHA delay_color= (255,255,255)
rect=(774,634,797,657) angular_velocity=4}//WHITE

//KUNAI

ShotData{ id=1592 render=ALPHA delay_color= (255,255,255)
rect=(689,53,701,73)}//RED
ShotData{ id=1593 render=ALPHA delay_color= (255,255,255)
rect=(701,53,713,73)}//MAGENTA
ShotData{ id=1594 render=ALPHA delay_color= (255,255,255)
rect=(713,53,725,73)}//PURPLE
ShotData{ id=1595 render=ALPHA delay_color= (255,255,255)
rect=(725,53,737,73)}//BLUE
ShotData{ id=1596 render=ALPHA delay_color= (255,255,255)
rect=(737,53,749,73)}//AQUA
```

```
ShotData{ id=1597 render=ALPHA delay_color= (255,255,255)
rect=(749,53,761,73)}//MINT
ShotData{ id=1598 render=ALPHA delay_color= (255,255,255)
rect=(761,53,773,73)}//GREEN
ShotData{ id=1599 render=ALPHA delay_color= (255,255,255)
rect=(773,53,785,73)}//YELLOW
ShotData{ id=1600 render=ALPHA delay_color= (255,255,255)
rect=(785,53,797,73)}//ORANGE
ShotData{ id=1601 render=ALPHA delay_color= (255,255,255)
rect=(797,53,809,73)}//WHITE
ShotData{ id=1602 render=ALPHA delay_color= (255,255,255)
rect=(809,53,821,73)}//BLACK

//ICICLE

ShotData{ id=1603 render=ALPHA delay_color= (255,255,255)
rect=(689,23,701,45)}//RED
ShotData{ id=1604 render=ALPHA delay_color= (255,255,255)
rect=(701,23,713,45)}//MAGENTA
ShotData{ id=1605 render=ALPHA delay_color= (255,255,255)
rect=(713,23,725,45)}//PURPLE
ShotData{ id=1606 render=ALPHA delay_color= (255,255,255)
rect=(725,23,737,45)}//BLUE
ShotData{ id=1607 render=ALPHA delay_color= (255,255,255)
rect=(737,23,749,45)}//AQUA
ShotData{ id=1608 render=ALPHA delay_color= (255,255,255)
rect=(749,23,761,45)}//MINT
ShotData{ id=1609 render=ALPHA delay_color= (255,255,255)
rect=(761,23,773,45)}//GREEN
ShotData{ id=1610 render=ALPHA delay_color= (255,255,255)
rect=(773,23,785,45)}//YELLOW
ShotData{ id=1611 render=ALPHA delay_color= (255,255,255)
rect=(785,23,797,45)}//ORANGE
ShotData{ id=1612 render=ALPHA delay_color= (255,255,255)
rect=(797,23,809,45)}//WHITE
ShotData{ id=1613 render=ALPHA delay_color= (255,255,255)
rect=(809,23,821,45)}//BLACK

//PELLET

ShotData{ id=1614 render=ALPHA delay_color= (255,255,255)
rect=(689,671,701,688)}//RED
ShotData{ id=1615 render=ALPHA delay_color= (255,255,255)
rect=(701,671,713,688)}//MAGENTA
ShotData{ id=1616 render=ALPHA delay_color= (255,255,255)
rect=(713,671,725,688)}//PURPLE
```

```
ShotData{ id=1617 render=ALPHA delay_color= (255,255,255)
rect=(725,671,737,688)}//BLUE
ShotData{ id=1618 render=ALPHA delay_color= (255,255,255)
rect=(737,671,749,688)}//AQUA
ShotData{ id=1619 render=ALPHA delay_color= (255,255,255)
rect=(749,671,761,688)}//MINT
ShotData{ id=1620 render=ALPHA delay_color= (255,255,255)
rect=(761,671,773,688)}//GREEN
ShotData{ id=1621 render=ALPHA delay_color= (255,255,255)
rect=(773,671,785,688)}//YELLOW
ShotData{ id=1622 render=ALPHA delay_color= (255,255,255)
rect=(785,671,797,688)}//ORANGE
ShotData{ id=1623 render=ALPHA delay_color= (255,255,255)
rect=(797,671,809,688)}//WHITE
ShotData{ id=1624 render=ALPHA delay_color= (255,255,255)
rect=(809,671,821,688)}//BLACK

//D_PELLET

ShotData{ id=1625 render=ALPHA delay_color= (255,255,255)
rect=(689,693,701,710)}//RED
ShotData{ id=1626 render=ALPHA delay_color= (255,255,255)
rect=(701,693,713,710)}//MAGENTA
ShotData{ id=1627 render=ALPHA delay_color= (255,255,255)
rect=(713,693,725,710)}//PURPLE
ShotData{ id=1628 render=ALPHA delay_color= (255,255,255)
rect=(725,693,737,710)}//BLUE
ShotData{ id=1629 render=ALPHA delay_color= (255,255,255)
rect=(737,693,749,710)}//AQUA
ShotData{ id=1630 render=ALPHA delay_color= (255,255,255)
rect=(749,693,761,710)}//MINT
ShotData{ id=1631 render=ALPHA delay_color= (255,255,255)
rect=(761,693,773,710)}//GREEN
ShotData{ id=1632 render=ALPHA delay_color= (255,255,255)
rect=(773,693,785,710)}//YELLOW
ShotData{ id=1633 render=ALPHA delay_color= (255,255,255)
rect=(785,693,797,710)}//ORANGE
ShotData{ id=1634 render=ALPHA delay_color= (255,255,255)
rect=(797,693,809,710)}//WHITE
ShotData{ id=1635 render=ALPHA delay_color= (255,255,255)
rect=(809,693,821,710)}//BLACK

//LARGE

ShotData{ id=1636 render=ALPHA delay_color= (255,255,255)
rect=(128,1030,161,1063) angular_velocity=1}//BLACK
```

```
ShotData{ id=1637 render=ALPHA delay_color= (255,255,255)
rect=(161,1030,194,1063) angular_velocity=-1}//RED
ShotData{ id=1638 render=ALPHA delay_color= (255,255,255)
rect=(194,1030,227,1063) angular_velocity=1}//MAGENTA
ShotData{ id=1639 render=ALPHA delay_color= (255,255,255)
rect=(227,1030,260,1063) angular_velocity=-1}//BLUE
ShotData{ id=1640 render=ALPHA delay_color= (255,255,255)
rect=(260,1030,293,1063) angular_velocity=1}//CYAN
ShotData{ id=1641 render=ALPHA delay_color= (255,255,255)
rect=(293,1030,326,1063) angular_velocity=-1}//AQUA
ShotData{ id=1642 render=ALPHA delay_color= (255,255,255)
rect=(326,1030,359,1063) angular_velocity=1}//MINT
ShotData{ id=1643 render=ALPHA delay_color= (255,255,255)
rect=(359,1030,392,1063) angular_velocity=-1}//GREEN
ShotData{ id=1644 render=ALPHA delay_color= (255,255,255)
rect=(392,1030,425,1063) angular_velocity=1}//YELLOW
ShotData{ id=1645 render=ALPHA delay_color= (255,255,255)
rect=(425,1030,458,1063) angular_velocity=-1}//ORANGE

//D_LARGE

ShotData{ id=1646 render=ALPHA delay_color= (255,255,255)
rect=(128,1030,161,1063) angular_velocity=-1}//BLACK
ShotData{ id=1647 render=ALPHA delay_color= (255,255,255)
rect=(161,1030,194,1063) angular_velocity=1}//RED
ShotData{ id=1648 render=ALPHA delay_color= (255,255,255)
rect=(194,1030,227,1063) angular_velocity=-1}//MAGENTA
ShotData{ id=1649 render=ALPHA delay_color= (255,255,255)
rect=(227,1030,260,1063) angular_velocity=1}//BLUE
ShotData{ id=1650 render=ALPHA delay_color= (255,255,255)
rect=(260,1030,293,1063) angular_velocity=-1}//CYAN
ShotData{ id=1651 render=ALPHA delay_color= (255,255,255)
rect=(293,1030,326,1063) angular_velocity=1}//AQUA
ShotData{ id=1652 render=ALPHA delay_color= (255,255,255)
rect=(326,1030,359,1063) angular_velocity=-1}//MINT
ShotData{ id=1653 render=ALPHA delay_color= (255,255,255)
rect=(359,1030,392,1063) angular_velocity=1}//GREEN
ShotData{ id=1654 render=ALPHA delay_color= (255,255,255)
rect=(392,1030,425,1063) angular_velocity=-1}//YELLOW
ShotData{ id=1655 render=ALPHA delay_color= (255,255,255)
rect=(425,1030,458,1063) angular_velocity=1}//ORANGE

//GLOW

ShotData{ id=1656 render=ALPHA delay_color= (255,255,255)
rect=(289,852,348,912) angular_velocity=-1}//RED
```

```
ShotData{ id=1657 render=ALPHA delay_color= (255,255,255)
rect=(348,852,407,912) angular_velocity=1}//PURPLE
ShotData{ id=1658 render=ALPHA delay_color= (255,255,255)
rect=(407,852,466,912) angular_velocity=-1}//BLUE
ShotData{ id=1659 render=ALPHA delay_color= (255,255,255)
rect=(289,912,348,972) angular_velocity=1}//AQUA
ShotData{ id=1660 render=ALPHA delay_color= (255,255,255)
rect=(348,912,407,972) angular_velocity=-1}//GREEN
ShotData{ id=1661 render=ALPHA delay_color= (255,255,255)
rect=(407,912,466,972) angular_velocity=1}//YELLOW

//FIREBALL

//RED
ShotData{ id=1662 render=ALPHA delay_color= (255,255,255)
AnimationData{
               animation_data=(6,553,665,579,710)
               animation_data=(6,579,665,605,710)
               animation_data=(6,605,665,632,710)
               animation_data=(6,632,665,659,710)
        }
}

//MAGENTA
ShotData{ id=1663 render=ALPHA delay_color= (255,255,255)
AnimationData{
               animation_data=(6,553,710,579,752)
               animation_data=(6,579,710,605,752)
               animation_data=(6,605,710,632,752)
               animation_data=(6,632,710,659,752)
        }
}

//PURPLE
ShotData{ id=1664 render=ALPHA delay_color= (255,255,255)
AnimationData{
               animation_data=(6,553,752,579,793)
               animation_data=(6,579,752,605,793)
               animation_data=(6,605,752,632,793)
               animation_data=(6,632,752,659,793)
        }
}

//BLUE
ShotData{ id=1665 render=ALPHA delay_color= (255,255,255)
AnimationData{
```

```
                animation_data=(6,553,793,579,833)
                animation_data=(6,579,793,605,833)
                animation_data=(6,605,793,632,833)
                animation_data=(6,632,793,659,833)
        }
}

//CYAN
ShotData{ id=1666 render=ALPHA delay_color= (255,255,255)
AnimationData{
                animation_data=(6,553,833,579,874)
                animation_data=(6,579,833,605,874)
                animation_data=(6,605,833,632,874)
                animation_data=(6,632,833,659,874)
        }
}

//AQUA
ShotData{ id=1667 render=ALPHA delay_color= (255,255,255)
AnimationData{
                animation_data=(6,553,874,579,916)
                animation_data=(6,579,874,605,916)
                animation_data=(6,605,874,632,916)
                animation_data=(6,632,874,659,916)
        }
}

//MINT
ShotData{ id=1668 render=ALPHA delay_color= (255,255,255)
AnimationData{
                animation_data=(6,553,916,579,957)
                animation_data=(6,579,916,605,957)
                animation_data=(6,605,916,632,957)
                animation_data=(6,632,916,659,957)
        }
}

//GREEN
ShotData{ id=1669 render=ALPHA delay_color= (255,255,255)
AnimationData{
                animation_data=(6,553,957,579,1000)
                animation_data=(6,579,957,605,1000)
                animation_data=(6,605,957,632,1000)
                animation_data=(6,632,957,659,1000)
        }
}
```

```
//YELLOW
ShotData{ id=1670 render=ALPHA delay_color= (255,255,255)
AnimationData{
               animation_data=(6,553,1000,579,1043)
               animation_data=(6,579,1000,605,1043)
               animation_data=(6,605,1000,632,1043)
               animation_data=(6,632,1000,659,1043)
        }
}

//ORANGE
ShotData{ id=1671 render=ALPHA delay_color= (255,255,255)
AnimationData{
               animation_data=(6,553,1043,579,1084)
               animation_data=(6,579,1043,605,1084)
               animation_data=(6,605,1043,632,1084)
               animation_data=(6,632,1043,659,1084)
        }
}



//------------------------------------------------------------------------
-------------------

//LASERS

ShotData{ id=1503 render=ALPHA delay_color= (255,255,255)
rect=(88,355,101,370)}//RED
ShotData{ id=1504 render=ALPHA delay_color= (255,255,255)
rect=(102,355,115,370)}//MAGENTA
ShotData{ id=1505 render=ALPHA delay_color= (255,255,255)
rect=(116,355,129,370)}//PURPLE
ShotData{ id=1506 render=ALPHA delay_color= (255,255,255)
rect=(130,355,143,370)}//BLUE
ShotData{ id=1507 render=ALPHA delay_color= (255,255,255)
rect=(144,355,157,370)}//AQUA
ShotData{ id=1508 render=ALPHA delay_color= (255,255,255)
rect=(158,355,171,370)}//MINT
ShotData{ id=1509 render=ALPHA delay_color= (255,255,255)
rect=(172,355,185,370)}//GREEN
ShotData{ id=1510 render=ALPHA delay_color= (255,255,255)
rect=(186,355,199,370)}//YELLOW
ShotData{ id=1511 render=ALPHA delay_color= (255,255,255)
rect=(200,355,213,370)}//ORANGE
```

```
ShotData{ id=1512 render=ALPHA delay_color= (255,255,255)
rect=(214,355,227,370)}//WHITE

//CUEVELASER

ShotData{ id=1513 render=ALPHA delay_color= (255,255,255)
rect=(588,3,598,74)}//RED
```

KyunBullet_Const.txt

```
/************************************************************************
***
*               Defines variables for easier access to bullet IDs and
loads                 *
*               the shotsheet into the system.
                                          *
*
                                                                        *
*               For ADD_ARGB versions of the shots, use the regular ID
plus 500.   *
*
                                                                        *
*               NOTE: Not every bullet ID has a variable to represent it.
                  *
************************************************************************
**/
local {
        let CSD = GetCurrentScriptDirectory;
        let path = CSD ~ "KyunBullet.txt";
        LoadEnemyShotData(path);
}

// Bone ---------------------------------
let BONE_WHITE = 1101;

// Artefakt ---------------------------------
let ARTEFAKT_YELLOW_GREEN      = 1102;

// light Mamizou Birds ---------------------------------
let BIRD_W_GREEN       = 1103;
let BIRD_W_BLUE= 1104;
let BIRD_W_YELLOW      = 1105;
let BIRD_W_RED = 1106;

// light Mamizou Birds ---------------------------------
let BIRD_B_GREEN       = 1107;
let BIRD_B_BLUE= 1108;
let BIRD_B_YELLOW      = 1109;
let BIRD_B_RED = 1110;

// Butterflys ---------------------------------
let FLY_BLACK  = 1111;
let FLY_RED    = 1112;
let FLY_PURPLE = 1113;
let FLY_BLUE   = 1114;
let FLY_AQUA   = 1115;
let FLY_GREEN  = 1116;
let FLY_YELLOW = 1117;
let FLY_ORANGE = 1118;

//Black Triangle ---------------------------------
let TRIANGLE1_BLACK    = 1119;
let TRIANGLE1_RED      = 1120;
let TRIANGLE1_PURPLE   = 1121;
let TRIANGLE1_PINK     = 1122;
let TRIANGLE1_BLUE     = 1123;
let TRIANGLE1_AQUA     = 1124;
let TRIANGLE1_GREEN    = 1125;
let TRIANGLE1_YELLOW   = 1126;
let TRIANGLE1_ORANGE   = 1127;
let TRIANGLE1_WHITE    = 1128;


//White Triangle ---------------------------------
let TRIANGLE2_BLACK    = 1129;
let TRIANGLE2_RED      = 1130;
let TRIANGLE2_PURPLE   = 1131;
let TRIANGLE2_PINK     = 1132;
let TRIANGLE2_BLUE     = 1133;
let TRIANGLE2_AQUA     = 1134;
let TRIANGLE2_GREEN    = 1135;
let TRIANGLE2_YELLOW   = 1136;
let TRIANGLE2_ORANGE   = 1137;
let TRIANGLE2_WHITE    = 1138;

//Magic Circle ---------------------------------
let MAGIC_BLACK        = 1139;
let MAGIC_RED  = 1140;
let MAGIC_PURPLE       = 1141;
let MAGIC_BLUE = 1142;
let MAGIC_AQUA = 1143;
let MAGIC_GREEN= 1144;
let MAGIC_YELLOW       = 1145;
let MAGIC_ORANGE       = 1146;

//Magic Circle Neo ---------------------------------
let N_MAGIC_RED= 1159;
let N_MAGIC_PINK       = 1160;
let N_MAGIC_BLUE       = 1161;
let N_MAGIC_AQUA       = 1156;
let N_MAGIC_DGREEN     = 1157;
let N_MAGIC_GREEN      = 1162;
let N_MAGIC_YELLOW     = 1158;
let N_MAGIC_WHITE      = 1155;

// IdoSpark------------------------------------------
let SPARK_RED          = 1147;
let SPARK_GREEN= 1148;
let SPARK_BLUE         = 1149;
let SPARK_YELLOW       = 1150;
let SPARK_PURPLE       = 1151;
let SPARK_AQUA = 1152;
let SPARK_ORANGE       = 1153;
let SPARK_WHITE= 1154;

// S Magic Star------------------------------------------
let MS_STAR_RED        = 1171;
let MS_STAR_PINK       = 1172;
let MS_STAR_BLUE               = 1173;
let MS_STAR_AQUA       = 1174;
let MS_STAR_GREEN      = 1175;
let MS_STAR_YELLOW     = 1176;
let MS_STAR_ORANGE     = 1177;
let MS_STAR_BLACK      = 1178;

// Shios Spear
        let SHIO_SPEAR = 1179;

// Gaia Mallet
        let GAIA_MALLET= 1180;

// Fumetsus

let A_FUMETSU_BW       = 1181;
let A_FUMETSU_RED      = 1182;
let A_FUMETSU_PURPLE   = 1183;
let A_FUMETSU_MAGENTA  = 1184;
let A_FUMETSU_BLUE     = 1185;
let A_FUMETSU_AQUA     = 1186;
let A_FUMETSU_GREEN    = 1187;
let A_FUMETSU_YELLOW   = 1188;
let A_FUMETSU_ORANGE   = 1189;

let B_FUMETSU_BW       = 1190;
let B_FUMETSU_RED      = 1191;
let B_FUMETSU_PURPLE   = 1192;
let B_FUMETSU_MAGENTA  = 1193;
let B_FUMETSU_BLUE     = 1194;
let B_FUMETSU_AQUA     = 1195;
let B_FUMETSU_GREEN    = 1196;
let B_FUMETSU_YELLOW   = 1197;
let B_FUMETSU_ORANGE   = 1198;

let C_FUMETSU_BW       = 1199;
let C_FUMETSU_AQUA     = 1200;
let C_FUMETSU_LGREEN   = 1201;
let C_FUMETSU_GREEN    = 1202;
let C_FUMETSU_YELLOW   = 1203;
let C_FUMETSU_ORANGE   = 1204;
let C_FUMETSU_PINK     = 1205;
let C_FUMETSU_VIOLET   = 1206;
let C_FUMETSU_CYAN     = 1207;

let D_FUMETSU_BW       = 1208;
let D_FUMETSU_MINT     = 1209;
let D_FUMETSU_YELLOW   = 1210;
let D_FUMETSU_LGREEN   = 1211;
let D_FUMETSU_ORANGE   = 1212;
let D_FUMETSU_RED      = 1213;
let D_FUMETSU_PURPLE   = 1214;
let D_FUMETSU_BLUE     = 1215;
let D_FUMETSU_AQUA     = 1216;

//ICESTAR

let ICESTAR_WHITE      = 1217;
let ICESTAR_RED        = 1218;
let ICESTAR_PURPLE     = 1219;
let ICESTAR_BLUE       = 1220;
let ICESTAR_AQUA       = 1221;
let ICESTAR_GREEN      = 1222;
let ICESTAR_YELLOW     = 1223;
let ICESTAR_ORANGE     = 1224;

//KBL_SMALL
```

```
let KBL_SMALL_WHITE    = 1225;
let KBL_SMALL_RED      = 1226;
let KBL_SMALL_PURPLE   = 1227;
let KBL_SMALL_BLUE     = 1228;
let KBL_SMALL_AQUA     = 1229;
let KBL_SMALL_DGREEN   = 1230;
let KBL_SMALL_GREEN    = 1231;
let KBL_SMALL_YELLOW   = 1232;
let KBL_SMALL_ORANGE   = 1233;
let KBL_SMALL_BLACK    = 1234;

//CHUINOTO

let CHUINOTO_WHITE             = 1235;
let CHUINOTO_RED               = 1236;
let CHUINOTO_MAGENTA   = 1237;
let CHUINOTO_PURPLE            = 1238;
let CHUINOTO_BLUE              = 1239;
let CHUINOTO_CYAN              = 1240;
let CHUINOTO_AQUA              = 1241;
let CHUINOTO_GREEN             = 1242;
let CHUINOTO_YELLOW            = 1243;
let CHUINOTO_ORANGE            = 1244;
let CHUINOTO_BLACK             = 1245;

//NICHUINOTO

let NICHUINOTO_WHITE   = 1246;
let NICHUINOTO_RED             = 1247;
let NICHUINOTO_MAGENTA         = 1248;
let NICHUINOTO_PURPLE  = 1249;
let NICHUINOTO_BLUE            = 1250;
let NICHUINOTO_CYAN            = 1251;
let NICHUINOTO_AQUA            = 1252;
let NICHUINOTO_GREEN   = 1253;
let NICHUINOTO_YELLOW  = 1254;
let NICHUINOTO_ORANGE  = 1255;
let NICHUINOTO_BLACK   = 1256;

//S_TEAR

let S_TEAR_RED         = 1257;
let S_TEAR_DRED        = 1258;
let S_TEAR_MAGENTA     = 1259;
let S_TEAR_BLUE        = 1260;
let S_TEAR_AQUA        = 1261;
let S_TEAR_GREEN       = 1262;
let S_TEAR_YELLOW      = 1263;
let S_TEAR_ORANGE      = 1264;

//TEAR

let TEAR_RED           = 1265;
let TEAR_DRED          = 1266;
let TEAR_MAGENTA       = 1267;
let TEAR_BLUE          = 1268;
let TEAR_AQUA          = 1269;
let TEAR_GREEN         = 1270;
let TEAR_YELLOW        = 1271;
let TEAR_ORANGE        = 1272;

//SPIKEHEAD

let SPIKEHEAD_RED              = 1273;
let SPIKEHEAD_DRED             = 1274;
let SPIKEHEAD_MAGENTA  = 1275;
let SPIKEHEAD_PURPLE   = 1276;
let SPIKEHEAD_BLUE             = 1277;
let SPIKEHEAD_CYAN             = 1278;
let SPIKEHEAD_AQUA             = 1279;
let SPIKEHEAD_GREEN            = 1280;
let SPIKEHEAD_YELLOW   = 1281;
let SPIKEHEAD_ORANGE   = 1282;

//DANGO

let DANGO_RED          = 1283;
let DANGO_PURPLE       = 1284;
let DANGO_BLUE         = 1285;
let DANGO_AQUA         = 1286;
let DANGO_GREEN        = 1287;
let DANGO_YELLOW       = 1288;
let DANGO_ORANGE       = 1289;

//LAVABALL

let LAVABALL_RED       = 1290;
let LAVABALL_BLUE      = 1291;
let LAVABALL_AQUA      = 1292;
let LAVABALL_GREEN     = 1293;
let LAVABALL_YELLOW    = 1294;
```

```
//S_ONMYOUGOKU

let S_ONMYOUGOKU_RED   = 1295;
let S_ONMYOUGOKU_PURPLE = 1296;
let S_ONMYOUGOKU_BLUE  = 1297;
let S_ONMYOUGOKU_AQUA  = 1298;
let S_ONMYOUGOKU_GREEN         = 1299;
let S_ONMYOUGOKU_YELLOW = 1300;

//ONMYOUGOKU

let ONMYOUGOKU_RED             = 1301;
let ONMYOUGOKU_PURPLE  = 1302;
let ONMYOUGOKU_BLUE    = 1303;
let ONMYOUGOKU_AQUA    = 1304;
let ONMYOUGOKU_GREEN   = 1305;
let ONMYOUGOKU_YELLOW  = 1306;

//SNOWFLAKE

let SNOWFLAKE_WHITE    = 1307;
let SNOWFLAKE_RED              = 1308;
let SNOWFLAKE_PURPLE   = 1309;
let SNOWFLAKE_BLUE             = 1310;
let SNOWFLAKE_AQUA             = 1311;
let SNOWFLAKE_GREEN    = 1312;
let SNOWFLAKE_YELLOW   = 1313;
let SNOWFLAKE_ORANGE   = 1314;

//SAKURA_BLOSSOM

let SAKURA_BLOSSOM_WHITE       = 1315;
let SAKURA_BLOSSOM_RED         = 1316;
let SAKURA_BLOSSOM_MAGENTA     = 1317;
let SAKURA_BLOSSOM_PURPLE      = 1318;
let SAKURA_BLOSSOM_BLUE        = 1319;
let SAKURA_BLOSSOM_AQUA        = 1320;
let SAKURA_BLOSSOM_GREEN       = 1321;
let SAKURA_BLOSSOM_YELLOW      = 1322;
let SAKURA_BLOSSOM_ORANGE      = 1323;
let SAKURA_BLOSSOM_BLACK       = 1324;

//SAKURA_ARROW

let SAKURA_ARROW_WHITE                 = 1325;
let SAKURA_ARROW_RED            = 1326;
let SAKURA_ARROW_MAGENTA        = 1327;
let SAKURA_ARROW_PURPLE         = 1328;
let SAKURA_ARROW_BLUE           = 1329;
let SAKURA_ARROW_AQUA           = 1330;
let SAKURA_ARROW_GREEN                  = 1331;
let SAKURA_ARROW_YELLOW         = 1332;
let SAKURA_ARROW_ORANGE         = 1333;
let SAKURA_ARROW_BLACK                  = 1334;

//ICEDIAMONDS

let ICEDIAMONDS_WHITE           = 1335;
let ICEDIAMONDS_LRED            = 1336;
let ICEDIAMONDS_DRED            = 1337;
let ICEDIAMONDS_MAGENTA         = 1338;
let ICEDIAMONDS_PURPLE                  = 1339;
let ICEDIAMONDS_BLUE            = 1340;
let ICEDIAMONDS_CYAN            = 1341;
let ICEDIAMONDS_AQUA            = 1342;
let ICEDIAMONDS_LGREEN                  = 1343;
let ICEDIAMONDS_GREEN           = 1344;
let ICEDIAMONDS_YELLOW                  = 1345;
let ICEDIAMONDS_ORANGE                  = 1346;

//STARBALL

let STARBALL_RED                = 1347;
let STARBALL_PURPLE             = 1348;
let STARBALL_BLUE               = 1349;
let STARBALL_AQUA               = 1350;
let STARBALL_GREEN              = 1351;
let STARBALL_YELLOW             = 1352;
let STARBALL_ORANGE             = 1353;

//STARRYORB

let A_STARRYORB_WHITE  = 1354;
let A_STARRYORB_RED             = 1355;
let A_STARRYORB_PURPLE = 1356;
let A_STARRYORB_BLUE   = 1357;
let A_STARRYORB_AQUA   = 1358;
let A_STARRYORB_GREEN  = 1359;
let A_STARRYORB_YELLOW = 1360;
let A_STARRYORB_ORANGE = 1361;
let B_STARRYORB_WHITE  = 1362;
let B_STARRYORB_RED            = 1363;
let B_STARRYORB_PURPLE = 1364;
let B_STARRYORB_BLUE   = 1365;
let B_STARRYORB_AQUA   = 1366;
let B_STARRYORB_GREEN  = 1367;
let B_STARRYORB_YELLOW = 1368;
let B_STARRYORB_ORANGE = 1369;

//YAJIRUSHI ARROW

let YAJIRUSHI_BLACK            = 1370;
let YAJIRUSHI_WHITE            = 1371;
let YAJIRUSHI_LRED             = 1372;
let YAJIRUSHI_RED              = 1373;
let YAJIRUSHI_LMAGENTA = 1374;
let YAJIRUSHI_MAGENTA  = 1375;
let YAJIRUSHI_LPURPLE  = 1376;
let YAJIRUSHI_PURPLE   = 1377;
let YAJIRUSHI_LBLUE            = 1378;
let YAJIRUSHI_BLUE             = 1379;
let YAJIRUSHI_LAQUA            = 1380;
let YAJIRUSHI_AQUA             = 1381;
let YAJIRUSHI_LGREEN   = 1382;
let YAJIRUSHI_GREEN            = 1383;
let YAJIRUSHI_LYELLOW  = 1384;
let YAJIRUSHI_YELLOW   = 1385;
let YAJIRUSHI_LORANGE  = 1386;
let YAJIRUSHI_ORANGE   = 1387;

//KAMELOTCANDLE(S)

let AA_KAMELOTCANDLE_RED             = 1388;
let AA_KAMELOTCANDLE_PURPLE          = 1389;
let AA_KAMELOTCANDLE_BLUE            = 1390;
let AA_KAMELOTCANDLE_AQUA            = 1391;
let AA_KAMELOTCANDLE_MINT            = 1392;
let AA_KAMELOTCANDLE_GREEN           = 1393;
let AA_KAMELOTCANDLE_YELLOW          = 1394;
let AA_KAMELOTCANDLE_ORANGE          = 1395;

let AB_KAMELOTCANDLE_RED             = 1396;
let AB_KAMELOTCANDLE_PURPLE          = 1397;
let AB_KAMELOTCANDLE_BLUE            = 1398;
let AB_KAMELOTCANDLE_AQUA            = 1399;
let AB_KAMELOTCANDLE_MINT            = 1400;
let AB_KAMELOTCANDLE_GREEN            = 1401;
let AB_KAMELOTCANDLE_YELLOW           = 1402;
let AB_KAMELOTCANDLE_ORANGE           = 1403;

let BA_KAMELOTCANDLE_RED              = 1404;
let BA_KAMELOTCANDLE_PURPLE           = 1405;
let BA_KAMELOTCANDLE_BLUE             = 1406;
let BA_KAMELOTCANDLE_AQUA             = 1407;
let BA_KAMELOTCANDLE_MINT             = 1408;
let BA_KAMELOTCANDLE_GREEN            = 1409;
let BA_KAMELOTCANDLE_YELLOW           = 1410;
let BA_KAMELOTCANDLE_ORANGE           = 1411;

let BB_KAMELOTCANDLE_RED              = 1412;
let BB_KAMELOTCANDLE_PURPLE           = 1413;
let BB_KAMELOTCANDLE_BLUE             = 1414;
let BB_KAMELOTCANDLE_AQUA             = 1415;
let BB_KAMELOTCANDLE_MINT             = 1416;
let BB_KAMELOTCANDLE_GREEN            = 1417;
let BB_KAMELOTCANDLE_YELLOW           = 1418;
let BB_KAMELOTCANDLE_ORANGE           = 1419;

let CA_KAMELOTCANDLE_MINT             = 1420;
let CA_KAMELOTCANDLE_GREEN            = 1421;
let CA_KAMELOTCANDLE_YELLOW           = 1422;
let CA_KAMELOTCANDLE_ORANGE           = 1423;
let CA_KAMELOTCANDLE_RED              = 1424;
let CA_KAMELOTCANDLE_PURPLE           = 1425;
let CA_KAMELOTCANDLE_BLUE             = 1426;
let CA_KAMELOTCANDLE_AQUA             = 1427;

let CB_KAMELOTCANDLE_MINT             = 1428;
let CB_KAMELOTCANDLE_GREEN            = 1429;
let CB_KAMELOTCANDLE_YELLOW           = 1430;
let CB_KAMELOTCANDLE_ORANGE           = 1431;
let CB_KAMELOTCANDLE_RED              = 1432;
let CB_KAMELOTCANDLE_PURPLE           = 1433;
let CB_KAMELOTCANDLE_BLUE             = 1434;
let CB_KAMELOTCANDLE_AQUA             = 1435;

let DA_KAMELOTCANDLE_MINT             = 1436;
let DA_KAMELOTCANDLE_GREEN     = 1437;
let DA_KAMELOTCANDLE_YELLOW           = 1438;
let DA_KAMELOTCANDLE_ORANGE           = 1439;
let DA_KAMELOTCANDLE_RED              = 1440;
let DA_KAMELOTCANDLE_PURPLE           = 1441;
let DA_KAMELOTCANDLE_BLUE             = 1442;
let DA_KAMELOTCANDLE_AQUA             = 1443;

let DB_KAMELOTCANDLE_MINT             = 1444;
let DB_KAMELOTCANDLE_GREEN            = 1445;
let DB_KAMELOTCANDLE_YELLOW           = 1446;
let DB_KAMELOTCANDLE_ORANGE           = 1447;
let DB_KAMELOTCANDLE_RED              = 1448;
let DB_KAMELOTCANDLE_PURPLE           = 1449;
let DB_KAMELOTCANDLE_BLUE             = 1450;
let DB_KAMELOTCANDLE_AQUA             = 1451;

//BLUBBLE

let BLUBBLE_WHITE               = 1452;
let BLUBBLE_RED                 = 1453;
let BLUBBLE_PURPLE              = 1454;
let BLUBBLE_BLUE                = 1455;
let BLUBBLE_AQUA                = 1456;
let BLUBBLE_GREEN               = 1457;
let BLUBBLE_YELLOW              = 1458;
let BLUBBLE_ORANGE              = 1459;

//S_BLUBBLE

let S_BLUBBLE_WHITE             = 1460;
let S_BLUBBLE_RED               = 1461;
let S_BLUBBLE_PURPLE   = 1462;
let S_BLUBBLE_BLUE              = 1463;
let S_BLUBBLE_AQUA              = 1464;
let S_BLUBBLE_GREEN             = 1465;
let S_BLUBBLE_YELLOW   = 1466;
let S_BLUBBLE_ORANGE   = 1467;

//STICK

let STICK_RED          = 1468;
let STICK_MAGENTA      = 1469;
let STICK_PURPLE       = 1470;
let STICK_BLUE         = 1471;
let STICK_AQUA         = 1472;
let STICK_MINT         = 1473;
let STICK_GREEN        = 1474;
let STICK_YELLOW       = 1475;
let STICK_ORANGE       = 1476;
```

```
//KOKORO

let KOKORO_RED                = 1477;
let KOKORO_DRED               = 1478;
let KOKORO_MAGENTA            = 1479;
let KOKORO_DMAGENTA           = 1480;
let KOKORO_BLUE               = 1481;
let KOKORO_DBLUE              = 1482;
let KOKORO_AQUA               = 1483;
let KOKORO_DAQUA              = 1484;
let KOKORO_GREEN              = 1485;
let KOKORO_DGREEN             = 1486;
let KOKORO_YELLOW             = 1487;
let KOKORO_DYELLOW            = 1488;

//FLOWER

let FLOWER_RED                = 1489;
let FLOWER_PURPLE             = 1490;
let FLOWER_BLUE               = 1491;
let FLOWER_AQUA               = 1492;
let FLOWER_GREEN              = 1493;
let FLOWER_YELLOW             = 1494;

//LEAF

let LEAF_WHITE                = 1495;
let LEAF_RED                  = 1496;
let LEAF_PURPLE               = 1497;
let LEAF_BLUE                 = 1498;
let LEAF_AQUA                 = 1499;
let LEAF_GREEN                = 1500;
let LEAF_YELLOW               = 1501;
let LEAF_ORANGE               = 1502;

//KNIFE

let KNIFE_WHITE               = 1519;
let KNIFE_RED                 = 1520;
let KNIFE_PURPLE              = 1521;
let KNIFE_BLUE                = 1522;
let KNIFE_CYAN                = 1523;
let KNIFE_AQUA                = 1524;
let KNIFE_GREEN               = 1525;
let KNIFE_YELLOW              = 1526;
let KNIFE_ORANGE              = 1527;
```

```
//C_KNIFE

let C_KNIFE_WHITE               = 1528;
let C_KNIFE_RED                 = 1529;
let C_KNIFE_PURPLE              = 1530;
let C_KNIFE_BLUE                = 1531;
let C_KNIFE_CYAN                = 1532;
let C_KNIFE_AQUA                = 1533;
let C_KNIFE_GREEN               = 1534;
let C_KNIFE_YELLOW              = 1535;
let C_KNIFE_ORANGE              = 1536;

//SWORD

let SWORD_WHITE                 = 1537;
let SWORD_RED                   = 1538;
let SWORD_PURPLE                = 1539;
let SWORD_BLUE                  = 1540;
let SWORD_CYAN                  = 1541;
let SWORD_AQUA                  = 1542;
let SWORD_GREEN                 = 1543;
let SWORD_YELLOW                = 1544;
let SWORD_ORANGE                = 1545;

//C_SWORD

let C_SWORD_WHITE               = 1546;
let C_SWORD_RED                 = 1547;
let C_SWORD_PURPLE              = 1548;
let C_SWORD_BLUE                = 1549;
let C_SWORD_CYAN                = 1550;
let C_SWORD_AQUA                = 1551;
let C_SWORD_GREEN               = 1552;
let C_SWORD_YELLOW              = 1553;
let C_SWORD_ORANGE              = 1554;

//SHURIKEN

let SHURIKEN_BLACK              = 1555;
let SHURIKEN_RED                = 1556;
let SHURIKEN_PURPLE             = 1557;
let SHURIKEN_BLUE               = 1558;
let SHURIKEN_AQUA               = 1559;
let SHURIKEN_GREEN              = 1560;
let SHURIKEN_YELLOW             = 1561;
let SHURIKEN_ORANGE            = 1562;

//S_SHURIKEN

let S_SHURIKEN_BLACK   = 1563;
let S_SHURIKEN_RED             = 1564;
let S_SHURIKEN_PURPLE  = 1565;
let S_SHURIKEN_BLUE            = 1566;
let S_SHURIKEN_AQUA            = 1567;
let S_SHURIKEN_GREEN   = 1568;
let S_SHURIKEN_YELLOW  = 1569;
let S_SHURIKEN_ORANGE  = 1570;

//S_PENTAGON

let S_PENTAGON_RED             = 1571;
let S_PENTAGON_PURPLE  = 1572;
let S_PENTAGON_BLUE            = 1573;
let S_PENTAGON_GREEN   = 1574;
let S_PENTAGON_YELLOW  = 1575;

//GEAR

let GEAR_RED           = 1576;
let GEAR_PURPLE        = 1577;
let GEAR_BLUE          = 1578;
let GEAR_AQUA          = 1579;
let GEAR_GREEN         = 1580;
let GEAR_YELLOW        = 1581;
let GEAR_ORANGE        = 1582;
let GEAR_WHITE         = 1583;

//S_GEAR

let S_GEAR_RED         = 1584;
let S_GEAR_PURPLE      = 1585;
let S_GEAR_BLUE        = 1586;
let S_GEAR_AQUA        = 1587;
let S_GEAR_GREEN       = 1588;
let S_GEAR_YELLOW      = 1589;
let S_GEAR_ORANGE      = 1590;
let S_GEAR_WHITE       = 1591;

//KUNAI

let KUNAI_RED          = 1592;
let KUNAI_MAGENTA      = 1593;
let KUNAI_PURPLE       = 1594;
let KUNAI_BLUE         = 1595;
let KUNAI_AQUA         = 1596;
let KUNAI_MINT         = 1597;
let KUNAI_GREEN        = 1598;
let KUNAI_YELLOW       = 1599;
let KUNAI_ORANGE       = 1600;
let KUNAI_WHITE        = 1601;
let KUNAI_BLACK        = 1602;

//ICICLE

let ICICLE_RED         = 1603;
let ICICLE_MAGENTA     = 1604;
let ICICLE_PURPLE      = 1605;
let ICICLE_BLUE        = 1606;
let ICICLE_AQUA        = 1607;
let ICICLE_MINT        = 1608;
let ICICLE_GREEN       = 1609;
let ICICLE_YELLOW      = 1610;
let ICICLE_ORANGE      = 1611;
let ICICLE_WHITE       = 1612;
let ICICLE_BLACK       = 1613;

//PELLET

let PELLET_RED         = 1614;
let PELLET_MAGENTA     = 1615;
let PELLET_PURPLE      = 1616;
let PELLET_BLUE        = 1617;
let PELLET_AQUA        = 1618;
let PELLET_MINT        = 1619;
let PELLET_GREEN       = 1620;
let PELLET_YELLOW      = 1621;
let PELLET_ORANGE      = 1622;
let PELLET_WHITE       = 1623;
let PELLET_BLACK       = 1624;

//D_PELLET

let D_PELLET_RED               = 1625;
let D_PELLET_MAGENTA   = 1626;
let D_PELLET_PURPLE            = 1627;
let D_PELLET_BLUE              = 1628;
let D_PELLET_AQUA              = 1629;
let D_PELLET_MINT              = 1630;
let D_PELLET_GREEN             = 1631;
let D_PELLET_YELLOW            = 1632;
let D_PELLET_ORANGE            = 1633;
let D_PELLET_WHITE             = 1634;
let D_PELLET_BLACK             = 1635;

//LARGE

let LARGE_BLACK        = 1636;
let LARGE_RED          = 1637;
let LARGE_MAGENTA      = 1638;
let LARGE_BLUE         = 1639;
let LARGE_CYAN         = 1640;
let LARGE_AQUA         = 1641;
let LARGE_MINT         = 1642;
let LARGE_GREEN        = 1643;
let LARGE_YELLOW       = 1644;
let LARGE_ORANGE       = 1645;

//D_LARGE

let D_LARGE_BLACK      = 1646;
let D_LARGE_RED        = 1647;
let D_LARGE_MAGENTA    = 1648;
let D_LARGE_BLUE       = 1649;
let D_LARGE_CYAN       = 1650;
let D_LARGE_AQUA       = 1651;
let D_LARGE_MINT       = 1652;
let D_LARGE_GREEN      = 1653;
let D_LARGE_YELLOW     = 1654;
let D_LARGE_ORANGE     = 1655;

//GLOW

let GLOW_RED    = 1656;
let GLOW_PURPLE= 1657;
let GLOW_BLUE   = 1658;
let GLOW_AQUA   = 1659;
let GLOW_GREEN  = 1660;
let GLOW_YELLOW= 1661;

//FIREBALL

let FIREBALL_RED               = 1662;
let FIREBALL_MAGENTA   = 1663;
let FIREBALL_PURPLE            = 1664;
let FIREBALL_BLUE              = 1665;
let FIREBALL_CYAN              = 1666;
let FIREBALL_AQUA              = 1667;
let FIREBALL_MINT              = 1668;
let FIREBALL_GREEN             = 1669;
let FIREBALL_YELLOW            = 1670;
let FIREBALL_ORANGE            = 1671;


//------------------------------------------------------------------------
---------------------------

//LASERS

let LASER_RED                  = 1503;
let LASER_MAGENTA              = 1504;
let LASER_PURPLE               = 1505;
let LASER_BLUE                 = 1506;
let LASER_AQUA                 = 1507;
let LASER_MINT                 = 1508;
let LASER_GREEN                = 1509;
let LASER_YELLOW               = 1510;
let LASER_ORANGE               = 1511;
let LASER_WHITE                = 1512;

//CUERVELASERS

let CUERVELASER_RED                    = 1503;
```

Package_ReplaySelectScene.txt

```
@Initialize
{
        SetAutoDeleteObject(true);

        TBackground();
        TReplaySelectScene();

        SetScriptResult("");
}

@MainLoop
{

        yield;
}
```

```
@Finalize
{
}

task TBackground()
{
        let dir = GetCurrentScriptDirectory();
        let pathTitle = dir ~ "img/TitleMenu.png";

        let objImage = ObjPrim_Create(OBJ_SPRITE_2D);
        Obj_SetRenderPriorityI(objImage, 20);
        ObjPrim_SetTexture(objImage, pathTitle);
        ObjSprite2D_SetSourceRect(objImage, 0, 0, 640, 480);
        ObjSprite2D_SetDestRect(objImage, 0, 0, 640, 480);
        ObjRender_SetAlpha(objImage, 64);

        let objText = ObjText_Create();
        ObjText_SetFontType(objText, "AsakuraSlab");
        ObjText_SetText(objText, "Repetir Selección");
        ObjText_SetFontSize(objText, 24);
        ObjText_SetFontBold(objText, true);
        ObjText_SetFontColorTop(objText, 255, 255, 51);
        ObjText_SetFontColorBottom(objText, 255, 255, 255);
        ObjText_SetFontBorderType(objText, BORDER_FULL);
        ObjText_SetFontBorderColor(objText,0, 0, 0);
        ObjText_SetFontBorderWidth(objText, 2);
        Obj_SetRenderPriorityI(objText, 30);
        ObjRender_SetX(objText, 16);
        ObjRender_SetY(objText, 16);
}

task TReplaySelectScene
{
        LoadReplayList();
        let listValidReplayIndex = GetValidReplayIndices();

        let cursorY = 0;
        let page = 0;
        let countMaxItem = length(listValidReplayIndex);
        let countItemPerPage = 20;
        let pageMax = trunc((countMaxItem - 1) / countItemPerPage);
        pageMax = max(pageMax, 0);
        let lastPageMaxCursorY = trunc(countMaxItem % countItemPerPage);
        if(countMaxItem % countItemPerPage == 0)
        {
                lastPageMaxCursorY = countItemPerPage;
        }

        task TMenuItem(let itemY)
        {
                let objText = CreateTextObject(32, 64 + 16 * itemY, 14,
"");
                ObjText_SetFontBorderWidth(objText, 1);
                let objSelect = CreateTextObject(32, 64 + 16 * itemY, 14,
"");
                ObjText_SetFontBorderWidth(objSelect, 1);
                ObjRender_SetBlendType(objSelect, BLEND_ADD_RGB);

                let oldPage = -1;
                loop
                {
                        if(page != oldPage)
                        {
                                let text = "";
                                let indexList = page * countItemPerPage +
itemY;
                                if(indexList < countMaxItem)
                                {
                                        let indexReplay =
listValidReplayIndex[indexList];
                                        if(IsValidReplayIndex(indexReplay))
                                        {
                                                text = rtos("00",
indexReplay) ~ " ";
                                                text = text ~ vtos("-8s",
GetReplayInfo(indexReplay, REPLAY_USER_NAME)) ~ " ";
                                                text = text ~
rtos("000000000000", GetReplayInfo(indexReplay, REPLAY_TOTAL_SCORE)) ~ "
";
                                                text = text ~ vtos("-8s",
GetReplayInfo(indexReplay, REPLAY_PLAYER_NAME)) ~ " ";
                                                text = text ~ vtos("03.2f",
GetReplayInfo(indexReplay, REPLAY_FPS_AVERAGE)) ~ "fps ";
                                                text = text ~
GetReplayInfo(indexReplay, REPLAY_DATE_TIME) ~ " ";

                                        }
                                }

                                ObjText_SetText(objText, text);
                                ObjText_SetText(objSelect, text);
                                oldPage = page;
                        }

                        if(page == pageMax && itemY >= lastPageMaxCursorY)
                        {
                                Obj_SetVisible(objText, false);
                                Obj_SetVisible(objSelect, false);
                        }
                        else
                        {
                                Obj_SetVisible(objText, true);
                                Obj_SetVisible(objSelect, itemY ==
cursorY);
                        }

                        yield;
                }
                Obj_Delete(objText);
                Obj_Delete(objSelect);
        }

        ascent(let iItem in 0 .. countItemPerPage)
        {
                TMenuItem(iItem);
        }

        while(GetVirtualKeyState(VK_OK) != KEY_FREE){yield;}

        let frameKeyHold = 0;
        loop
        {
                if(GetVirtualKeyState(VK_OK) == KEY_PULL)
                {
                        let indexList = page * countItemPerPage + cursorY;
                        if(indexList < countMaxItem)
                        {
                                let indexReplay =
listValidReplayIndex[indexList];
                                let pathReplay = GetReplayInfo(indexReplay,
REPLAY_FILE_PATH);
                                SetScriptResult(pathReplay);
                                CloseScript(GetOwnScriptID());
                                break;
                        }
                }
                if(GetVirtualKeyState(VK_CANCEL) == KEY_PULL)
                {
                        CloseScript(GetOwnScriptID());
                        break;
                }

                if(GetVirtualKeyState(VK_UP) == KEY_PUSH ||
GetVirtualKeyState(VK_UP) == KEY_HOLD)
                {
                        frameKeyHold++;
                        if(GetVirtualKeyState(VK_UP) == KEY_PUSH ||
                                frameKeyHold == 20 ||
                                 (frameKeyHold > 20 && (frameKeyHold % 5 ==
0)))
                        {
                                cursorY--;
                        }
                }
                else if(GetVirtualKeyState(VK_DOWN) == KEY_PUSH ||
GetVirtualKeyState(VK_DOWN) == KEY_HOLD)
                {
                        frameKeyHold++;
                        if(GetVirtualKeyState(VK_DOWN) == KEY_PUSH ||
                                frameKeyHold == 20 ||
                                 (frameKeyHold > 20 && (frameKeyHold % 5 ==
0)))
                        {
                                cursorY++;
                        }
                }
                else if(GetVirtualKeyState(VK_LEFT) == KEY_PUSH ||
GetVirtualKeyState(VK_LEFT) == KEY_HOLD)
                {
                        frameKeyHold++;
                        if(GetVirtualKeyState(VK_LEFT) == KEY_PUSH ||
                                frameKeyHold == 20 ||
                                 (frameKeyHold > 20 && (frameKeyHold % 5 ==
0)))
                        {
                                page--;
                        }
                }
                else if(GetVirtualKeyState(VK_RIGHT) == KEY_PUSH ||
GetVirtualKeyState(VK_RIGHT) == KEY_HOLD)
                {
                        frameKeyHold++;
                        if(GetVirtualKeyState(VK_RIGHT) == KEY_PUSH ||
                                frameKeyHold == 20 ||
                                 (frameKeyHold > 20 && (frameKeyHold % 5 ==
0)))
                        {
                                page++;
                        }
                }
                else
                {
                        frameKeyHold = 0;
                }

                if(page < 0)
                {
                        page = pageMax;
                }
                else if(page > pageMax)
                {
                        page = 0;
                }

                if(page != pageMax)
                {
                        if(cursorY < 0)
                        {
                                cursorY = countItemPerPage - 1;
                        }
                        else if(cursorY >= countItemPerPage)
                        {
                                cursorY = 0;
                        }
                }
                else
                {
                        if(cursorY < 0)
                        {
                                cursorY = lastPageMaxCursorY - 1;
                        }
                        else if(cursorY >= lastPageMaxCursorY)
                        {
                                cursorY = 0;
                        }
                }

                yield;
        }
}

function CreateTextObject(let mx, let my, let size, let text)
{
        let obj = ObjText_Create();
        ObjText_SetText(obj, text);
        ObjText_SetFontSize(obj, size);
        ObjText_SetFontBold(obj, true);
        ObjText_SetFontColorTop(obj, 51, 153, 255);
        ObjText_SetFontColorBottom(obj, 255, 255, 255);
        ObjText_SetFontBorderType(obj, BORDER_FULL);
        ObjText_SetFontBorderColor(obj, 0, 51, 102);
        ObjText_SetFontBorderWidth(obj, 2);
        Obj_SetRenderPriorityI(obj, 40);
        ObjRender_SetX(obj, mx);
        ObjRender_SetY(obj, my);
        return obj;
}
```

PrimitiveTest.dnh

```
//BY ULTIMA

task
Create2DObject(image,vertex,vertex_length,blend_type,priority,posY,alpha,
speed,colorR,colorG,colorB){
let vertexcount = vertex;
let radius = 550;
let divideby = radius;
let obj = ObjPrim_Create(OBJ_PRIMITIVE_2D);
ObjPrim_SetPrimitiveType(obj,PRIMITIVE_TRIANGLESTRIP);
ObjPrim_SetVertexCount(obj,vertexcount);
ObjPrim_SetTexture(obj,image);
ObjRender_SetColor(obj,colorR,colorG,colorB);

ObjRender_SetPosition(obj,GetStgFrameWidth/2,posY,00);
ascent(i in 0..vertexcount){
let indexvert = i*2;
let left = i*(512/3);
let angle = 360/(vertexcount/2-1)*i;
ObjPrim_SetVertexUVT(obj,indexvert,-left,0);
ObjPrim_SetVertexUVT(obj,indexvert+1,-left,512);
ObjPrim_SetVertexPosition(obj,indexvert+0,(0)*cos(angle),(0)*sin(angle),0
);
ObjPrim_SetVertexPosition(obj,indexvert+1,(radius)*cos(angle),(radius)*si
n(angle),0);
```

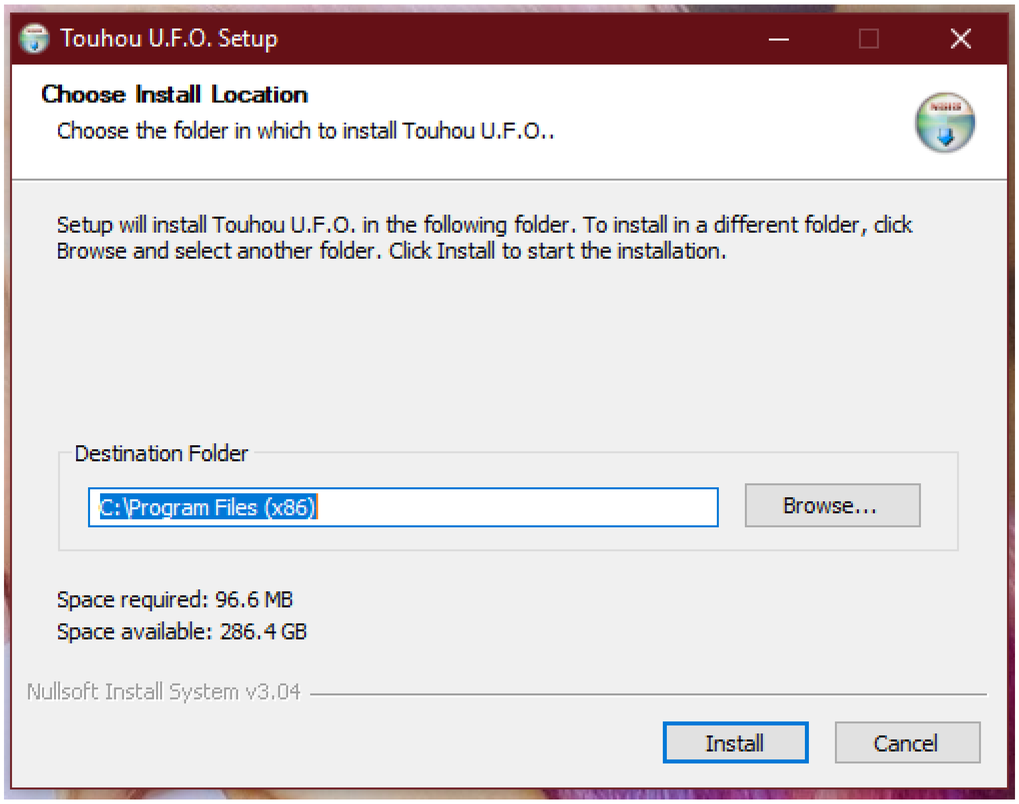
```
Obj_SetRenderPriorityI(obj,priority);
if(blend_type=="ADD"){ObjRender_SetBlendType(obj,BLEND_ADD_ARGB);}
if(blend_type=="SUB"){ObjRender_SetBlendType(obj,BLEND_SUBTRACT);}

//loop(1){yield;}
}
let mx = 0;
loop{
mx-=speed;
ascent(i in 0..vertexcount){
let indexvert = i*2;

let angle = 360/(vertexcount/2)*i;
ObjPrim_SetVertexUVT(obj,indexvert,128*cos(angle),0+mx);
ObjPrim_SetVertexUVT(obj,indexvert+1,128*cos(angle),vertex_length+mx);

}
ObjRender_SetAlpha(obj,alpha);
yield;
}
ObjRender_SetZWrite(obj,true);
ObjRender_SetZTest(obj,true);
ObjRender_SetAngleX(obj,90);
ObjRender_SetPosition(obj,0,0,00);
//return obj
}

task
Create2DObject2(image,vertex,vertex_length,blend_type,priority,posY,alpha
,speed,colorR,colorG,colorB){
let vertexcount = vertex;
let radius = 500;
let divideby = radius;
let obj = ObjPrim_Create(OBJ_PRIMITIVE_2D);
ObjPrim_SetPrimitiveType(obj,PRIMITIVE_TRIANGLESTRIP);
ObjPrim_SetVertexCount(obj,vertexcount);
ObjPrim_SetTexture(obj,image);
ObjRender_SetColor(obj,colorR,colorG,colorB);

ObjRender_SetPosition(obj,GetStgFrameWidth/2,posY,00);
ascent(i in 0..vertexcount){
let indexvert = i*2;
let left = i*(512/3);
let angle = 360/(vertexcount/2-1)*i;
ObjPrim_SetVertexUVT(obj,indexvert,-left,0);
ObjPrim_SetVertexUVT(obj,indexvert+1,-left,512);
ObjPrim_SetVertexPosition(obj,indexvert+0,(0)*cos(angle),(0)*sin(angle),0
);
ObjPrim_SetVertexPosition(obj,indexvert+1,(radius)*cos(angle),(radius)*si
n(angle),0);

Obj_SetRenderPriorityI(obj,priority);
if(blend_type=="ADD"){ObjRender_SetBlendType(obj,BLEND_ADD_ARGB);}
if(blend_type=="SUB"){ObjRender_SetBlendType(obj,BLEND_SUBTRACT);}

//loop(1){yield;}
}
let mx = 0;
loop{
mx-=speed;
ascent(i in 0..vertexcount){
let indexvert = i*2;

let angle = 360/(vertexcount/2)*i;
ObjPrim_SetVertexUVT(obj,indexvert,128*cos(angle),0+mx);
ObjPrim_SetVertexUVT(obj,indexvert+1,128*cos(angle),vertex_length+mx);

}
ObjRender_SetAlpha(obj,GetCommonData("BGAlpha",0));
yield;
}
ObjRender_SetZWrite(obj,true);
ObjRender_SetZTest(obj,true);
ObjRender_SetAngleX(obj,90);
ObjRender_SetPosition(obj,0,0,00);
//return obj
}
```

SELibrary.txt

```
let poof =  GetCurrentScriptDirectory~"./se/se_enep00_2.wav";
let explosion = GetCurrentScriptDirectory~"./se/se_enep01.wav";

let dam1 = GetCurrentScriptDirectory~"./se/se_damage00.wav";
let dam2 = GetCurrentScriptDirectory~"./se/se_damage01.wav";

let shot1 = GetCurrentScriptDirectory~"./se/se_tan00.wav";
let shot2 = GetCurrentScriptDirectory~"./se/se_tan01.wav";
let shot3 = GetCurrentScriptDirectory~"./se/se_tan02.wav";
let shot4 = GetCurrentScriptDirectory~"./se/se_tan03.wav";

let chime1 = GetCurrentScriptDirectory~"./se/se_kira00.wav";
let chime2 = GetCurrentScriptDirectory~"./se/se_kira01.wav";

let laser0 = GetCurrentScriptDirectory~"./se/se_lazer00.wav";
let laser1 = GetCurrentScriptDirectory~"./se/se_gun00.wav";
let laser2 = GetCurrentScriptDirectory~"./se/se_lazer01.wav";
let laser3 = GetCurrentScriptDirectory~"./se/Laser2.wav";

let lenlaser1 = GetCurrentScriptDirectory~"./se/lenlaser1.wav";
let lenlaser2 = GetCurrentScriptDirectory~"./se/lenlaser2.wav";
let lenshot1 = GetCurrentScriptDirectory~"./se/lenshot4.wav";
let lenshot2 = GetCurrentScriptDirectory~"./se/lenshot5.wav";

let lencharge1 = GetCurrentScriptDirectory~"./se/charge_2.wav";

let charge0 = GetCurrentScriptDirectory~"./se/se_ch00.wav";
let charge1 = GetCurrentScriptDirectory~"./se/se_ch02.wav";
let charge2 = GetCurrentScriptDirectory~"./se/se_power0.wav";
let charge3 = GetCurrentScriptDirectory~"./se/charge.wav";
let charge4 = GetCurrentScriptDirectory~"./se/power0.wav";
let charge5 = GetCurrentScriptDirectory~"./se/Charge2.wav";

let DED = GetCurrentScriptDirectory~"./se/se_pldead00.wav";
let DEDEND = GetCurrentScriptDirectory~"./se/se_pldead01.wav";

let explode = GetCurrentScriptDirectory~"./se/se_enep02.wav";
let explode2 = GetCurrentScriptDirectory~"./se/se_explode.wav";

let bubble = GetCurrentScriptDirectory~"./se/bubble.wav";

let boon0 = GetCurrentScriptDirectory~"./se/se_boon00.wav";
let boon1 = GetCurrentScriptDirectory~"./se/se_boon01.wav";

let compositespell = GetCurrentScriptDirectory~"./se/compositespell.wav";
let spellcard = GetCurrentScriptDirectory~"./se/se_cat00.wav";

let timeout1 = GetCurrentScriptDirectory~"./se/se_timeout.wav";
let timeout2 = GetCurrentScriptDirectory~"./se/se_timeout2.wav";

let Confirm = GetCurrentScriptDirectory~"./se/se_ok00.wav";
let Choose = GetCurrentScriptDirectory~"./se/se_select00.wav";
let ping = GetCurrentScriptDirectory~"./se/ping.wav";

let alert = GetCurrentScriptDirectory~"./se/se_life1.wav";
let hum = GetCurrentScriptDirectory~"./se/lenmecha04.wav";
let hop = GetCurrentScriptDirectory~"./se/se_lgodsget.wav";
let outerring = GetCurrentScriptDirectory~"./se/se_lgods2.wav";
let complete = GetCurrentScriptDirectory~"./se/se_cardget.wav";

let HMexplode1 = GetCurrentScriptDirectory~"./se/846.wav";
let HMexplode2 = GetCurrentScriptDirectory~"./se/847.wav";
let HMexplode3 = GetCurrentScriptDirectory~"./se/832.wav";
let HMexplode4 = GetCurrentScriptDirectory~"./se/firework_final.wav";

let ding = GetCurrentScriptDirectory~"./se/ding.wav";
let split = GetCurrentScriptDirectory~"./se/split.wav";
let drumroll = GetCurrentScriptDirectory~"./se/drumroll.wav";
let wham = GetCurrentScriptDirectory~"./se/thud.wav";

/////////BONUS///////////
let sumo = GetCurrentScriptDirectory~"./se/bonus/subaluwa.wav";
let glint = GetCurrentScriptDirectory~"./se/bonus/glint.wav";
let chaching = GetCurrentScriptDirectory~"./se/bonus/cha-ching.wav";
let eviltalk = GetCurrentScriptDirectory~"./se/bonus/eviltalk.wav";

task SE_Play(let path, let vl){
let seobj = ObjSound_Create;
ObjSound_Load(seobj,path);
ObjSound_Play(seobj);
ObjSound_SetVolumeRate(seobj,vl);
loop(120){yield;}
RemoveSound(path);
}
```

GUIA DE INSTALACION

Abrir el archivo th_dnh.exe para iniciar el instalador


Touhou U.F.O. Setup

Choose Install Location

Choose the folder in which to install Touhou U.F.O..

Setup will install Touhou U.F.O. in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

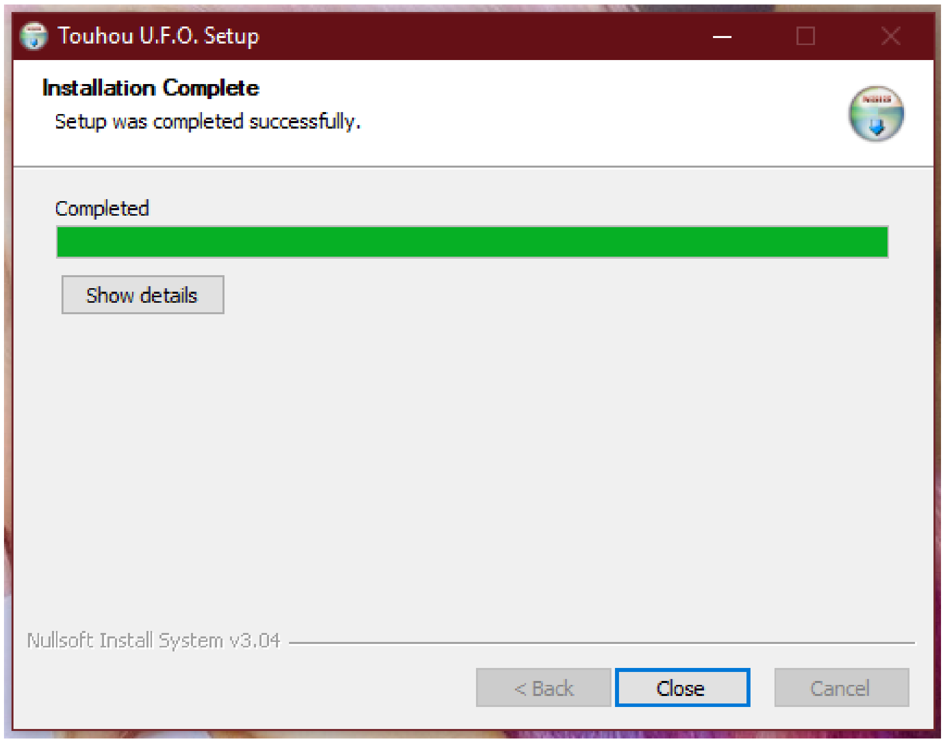
Destination Folder

C:\Program Files (x86)

Space required: 96.6 MB
Space available: 286.4 GB

Nullsoft Install System v3.04

Seleccionar la carpeta donde será instalado el programa y dar click en ***Install***



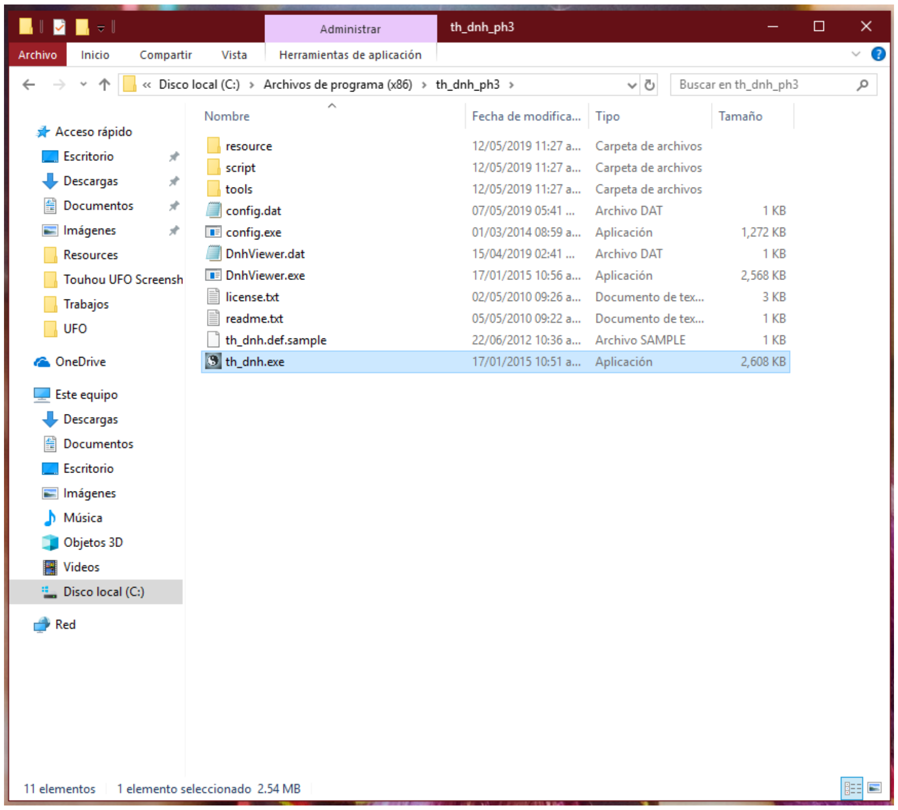
Installation Complete

Setup was completed successfully.

Completed

Nullsoft Install System v3.04

Una vez completada la instalación cerrar el instalador e ir a la carpeta donde se instalo el programa

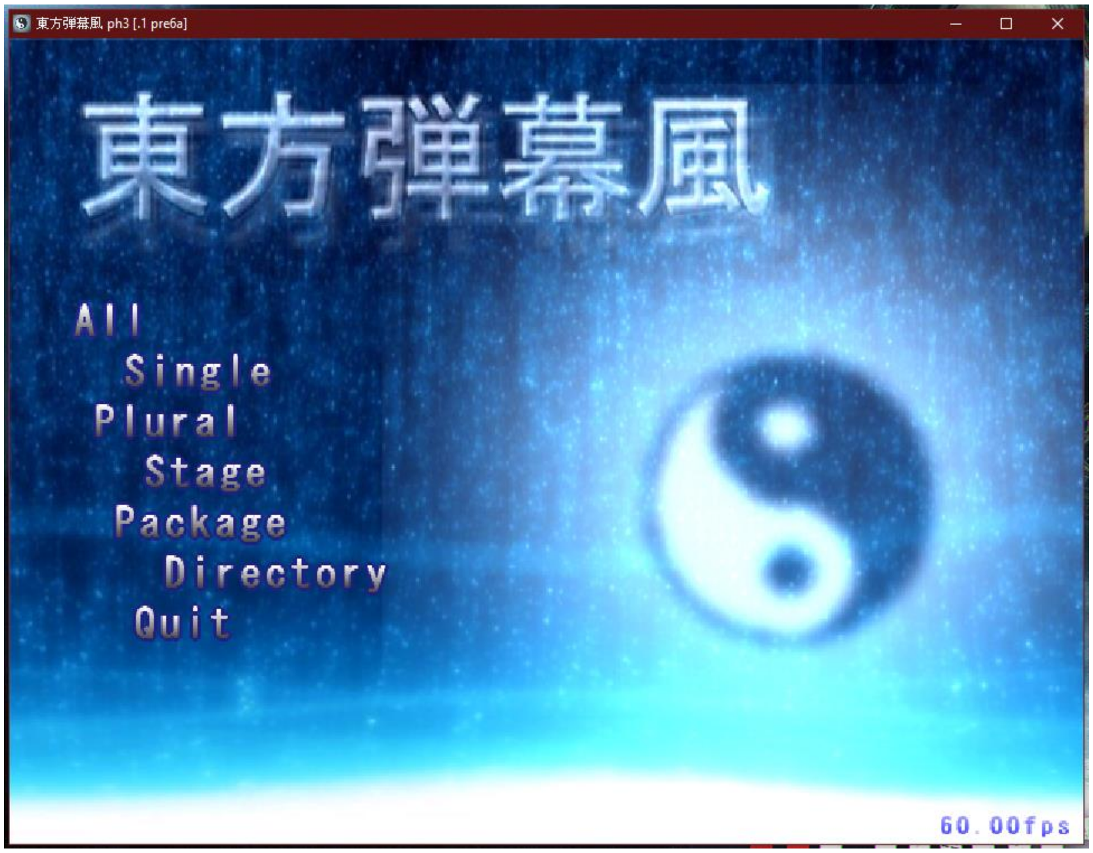


« Disco local (C:) › Archivos de programa (x86) › th_dnh_ph3 ›

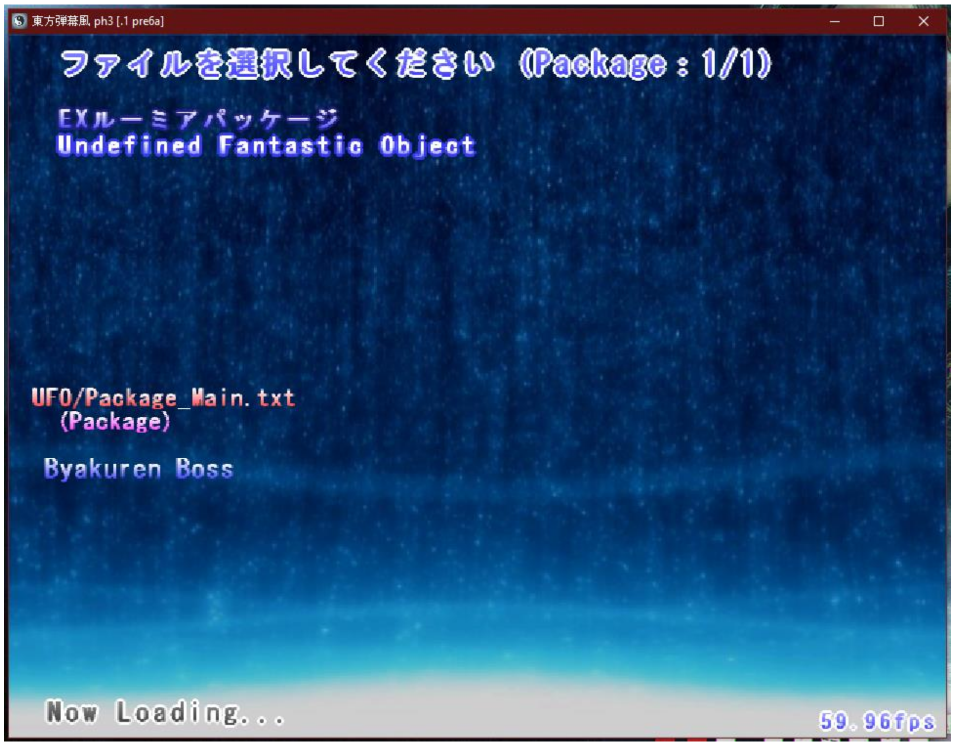
Nombre	Fecha de modifica...	Tipo	Tamaño
resource	12/05/2019 11:27 a...	Carpeta de archivos	
script	12/05/2019 11:27 a...	Carpeta de archivos	
tools	12/05/2019 11:27 a...	Carpeta de archivos	
config.dat	07/05/2019 05:41 ...	Archivo DAT	1 KB
config.exe	01/03/2014 08:59 a...	Aplicación	1,272 KB
DnhViewer.dat	15/04/2019 02:41 ...	Archivo DAT	1 KB
DnhViewer.exe	17/01/2015 10:56 a...	Aplicación	2,568 KB
license.txt	02/05/2010 09:26 a...	Documento de tex...	3 KB
readme.txt	05/05/2010 09:22 a...	Documento de tex...	1 KB
th_dnh.def.sample	22/06/2012 10:36 a...	Archivo SAMPLE	1 KB
th_dnh.exe	17/01/2015 10:51 a...	Aplicación	2,608 KB

11 elementos   1 elemento seleccionado 2.54 MB

Ejecutar **th_dnh.exe**

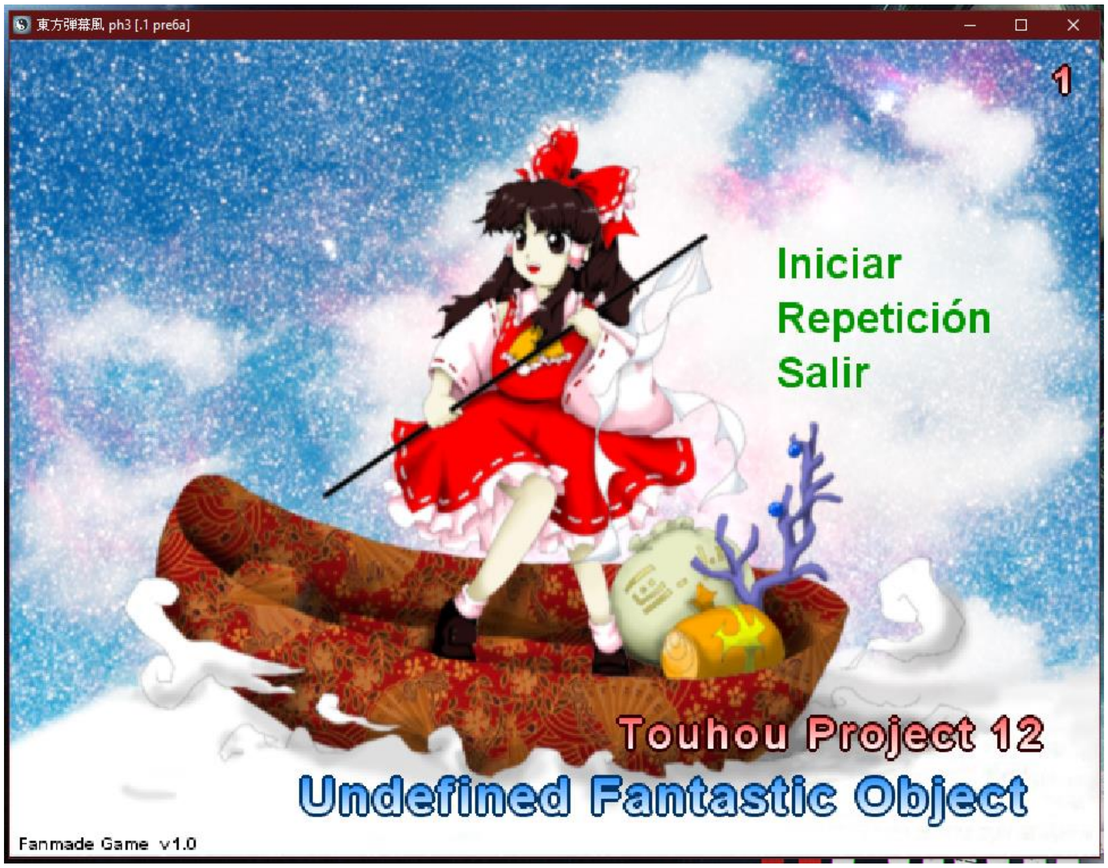


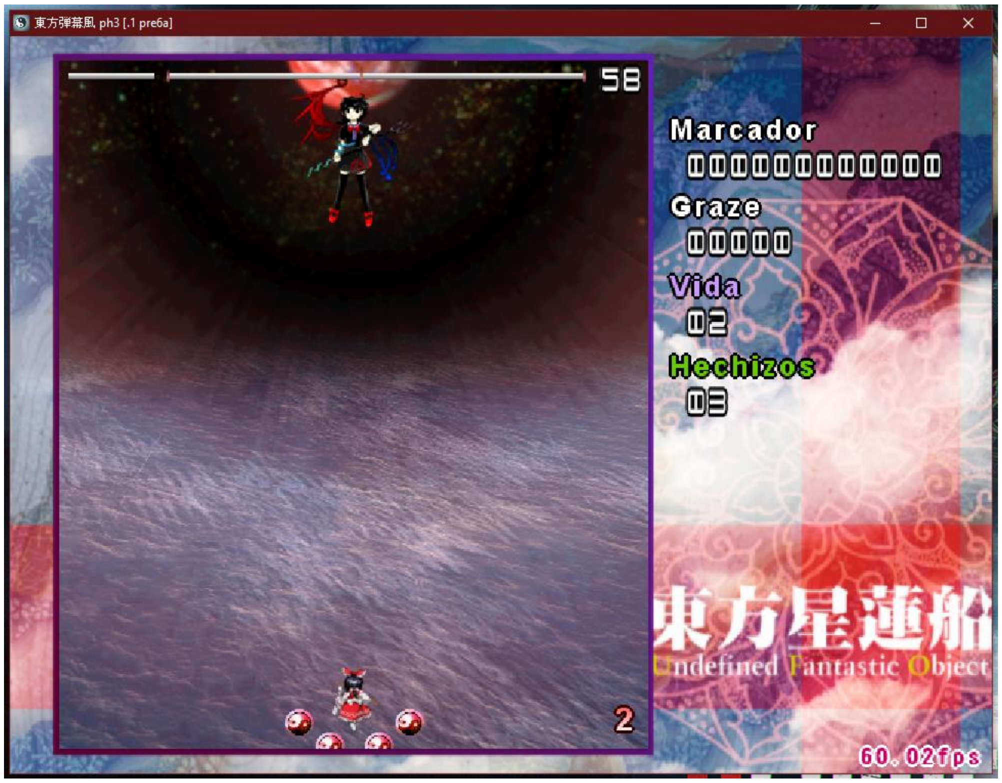
Dentro del menú ir a la opción *“**Package**”*

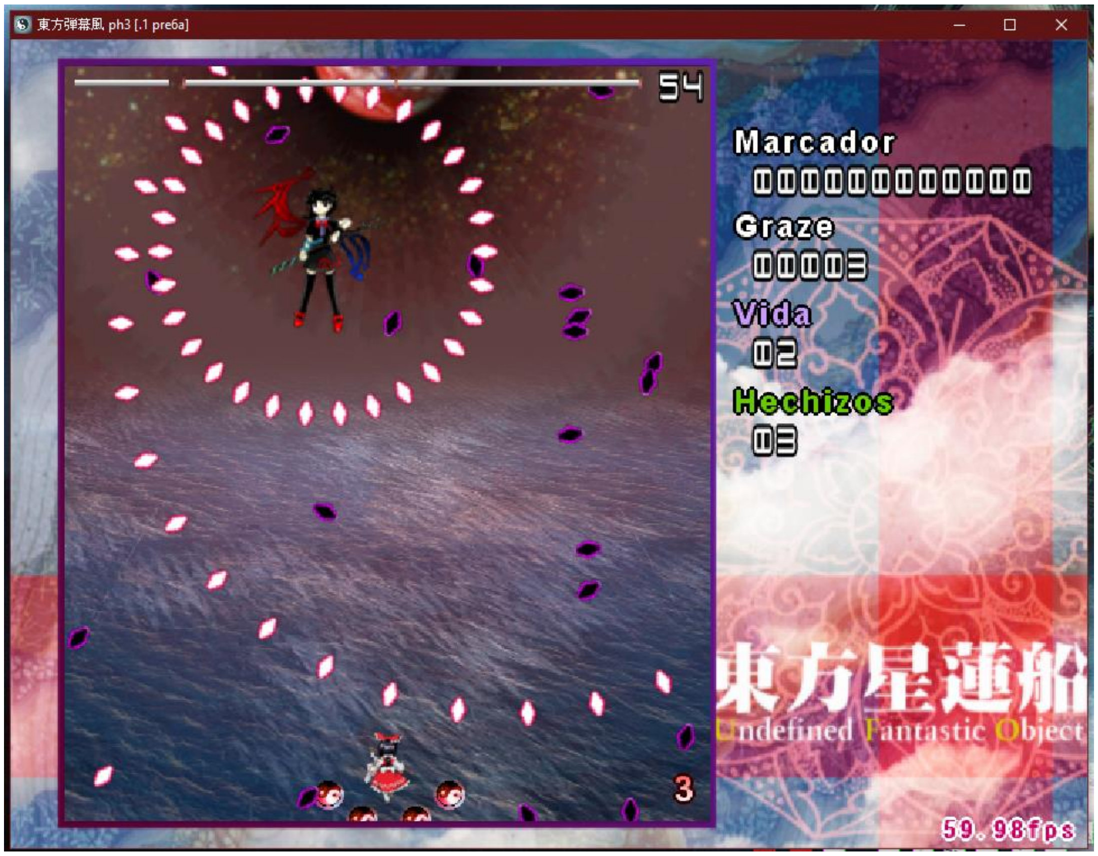

東方弾幕風 ph3 [.1 pre6a]
ファイルを選択してください (Package : 1/1)
EXルーミアパッケージ
Undefined Fantastic Object
UFO/Package_Main.txt
(Package)
Byakuren Boss
Now Loading...
59.96fps

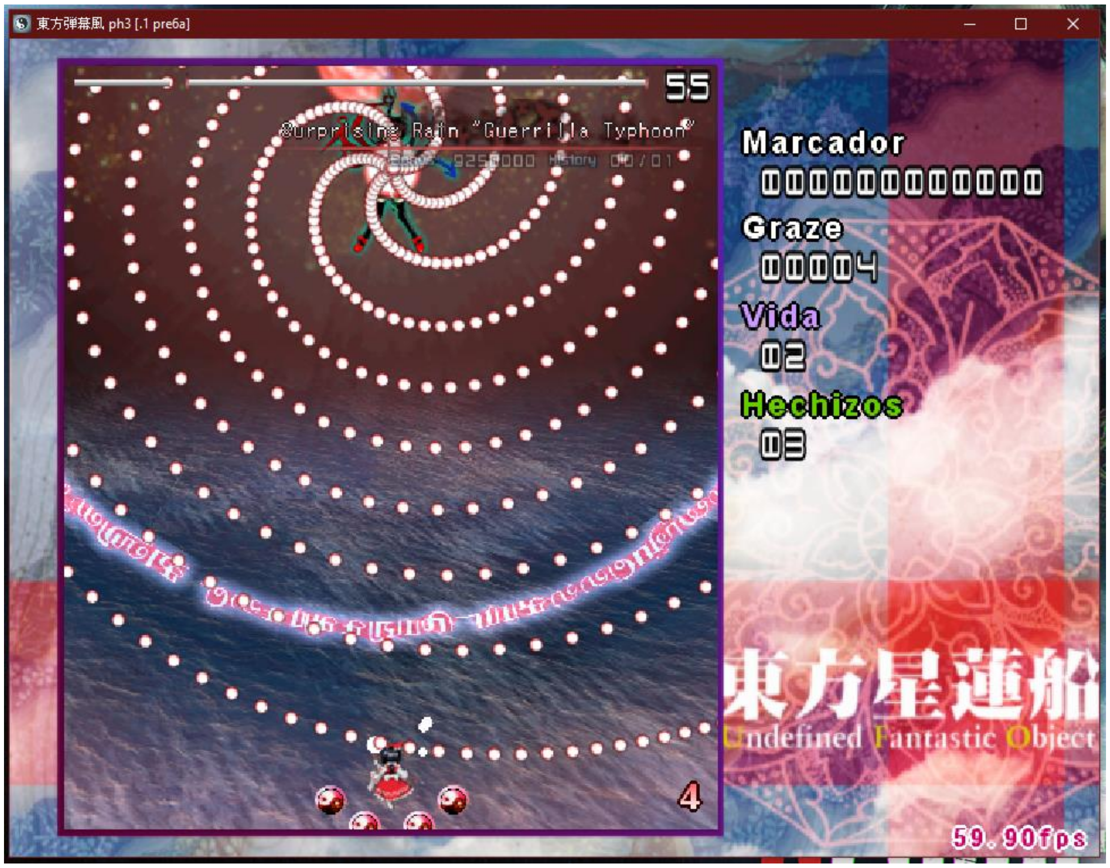
Dentro seleccionar ***Undefined Fantastic Object*** para iniciar la ejecución del juego

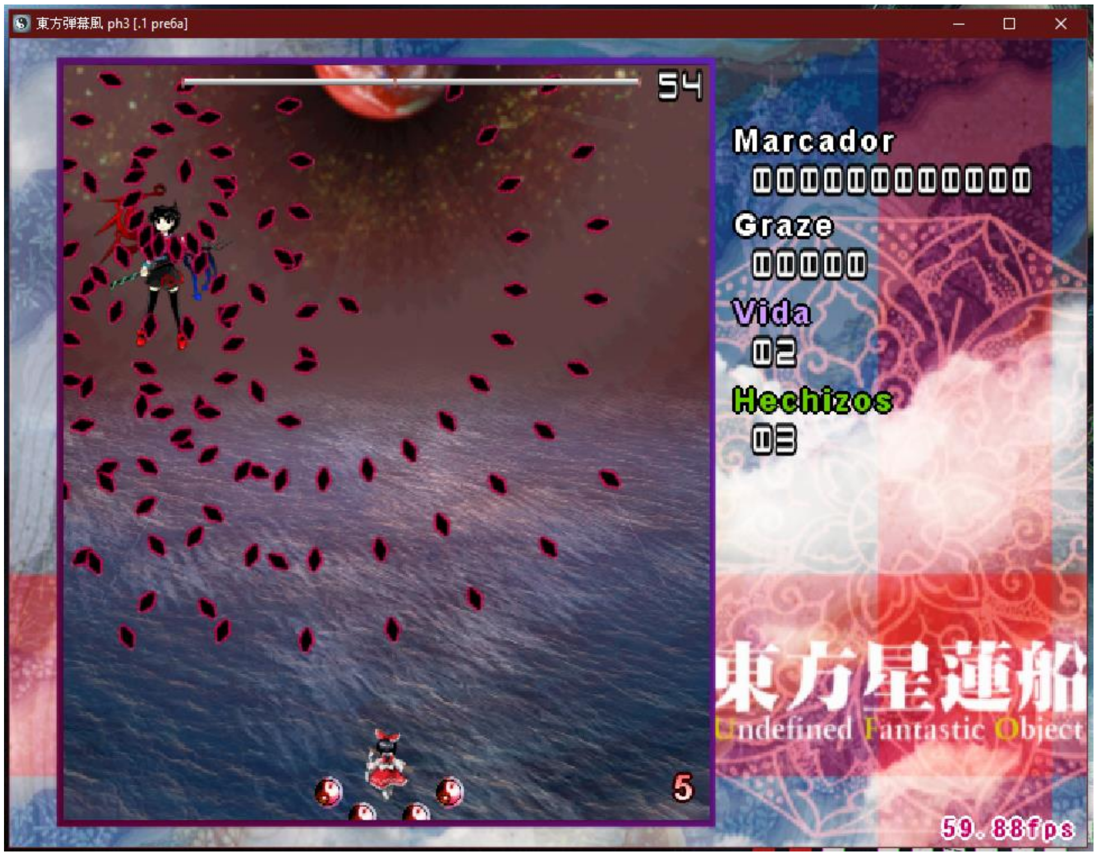
CAPTURAS DE PANTALLA

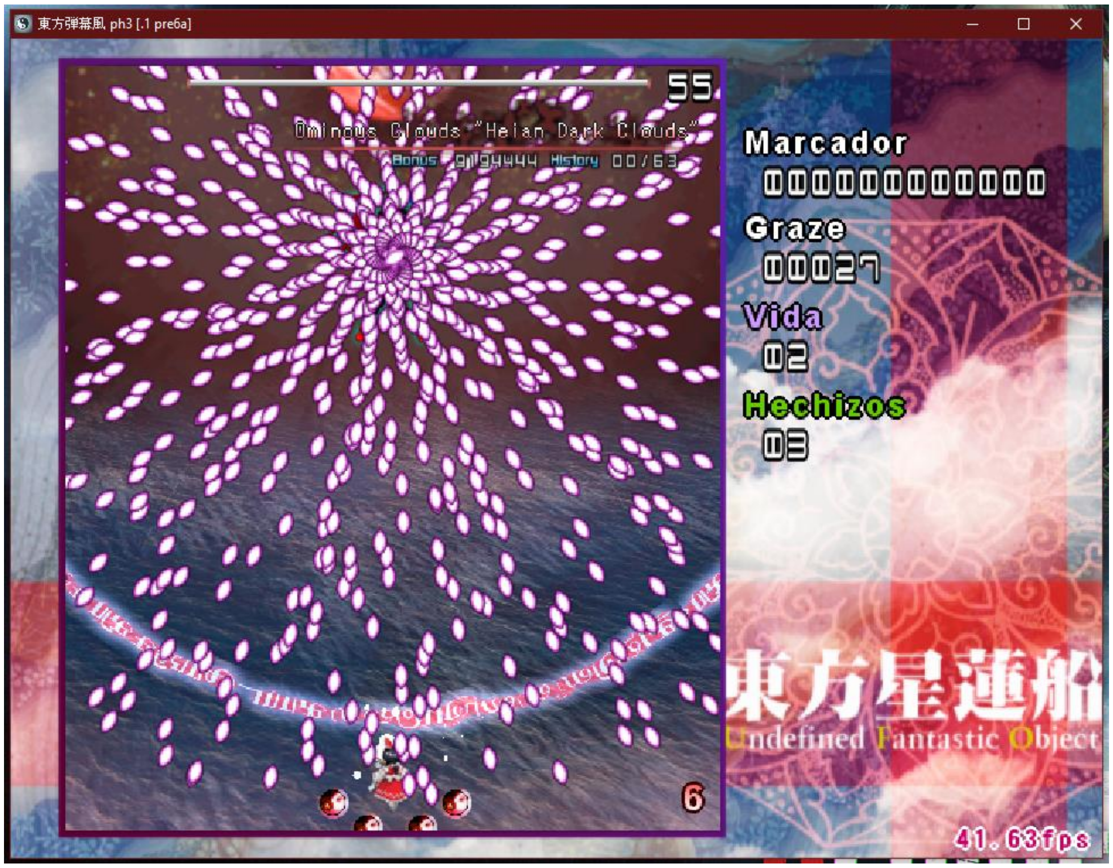


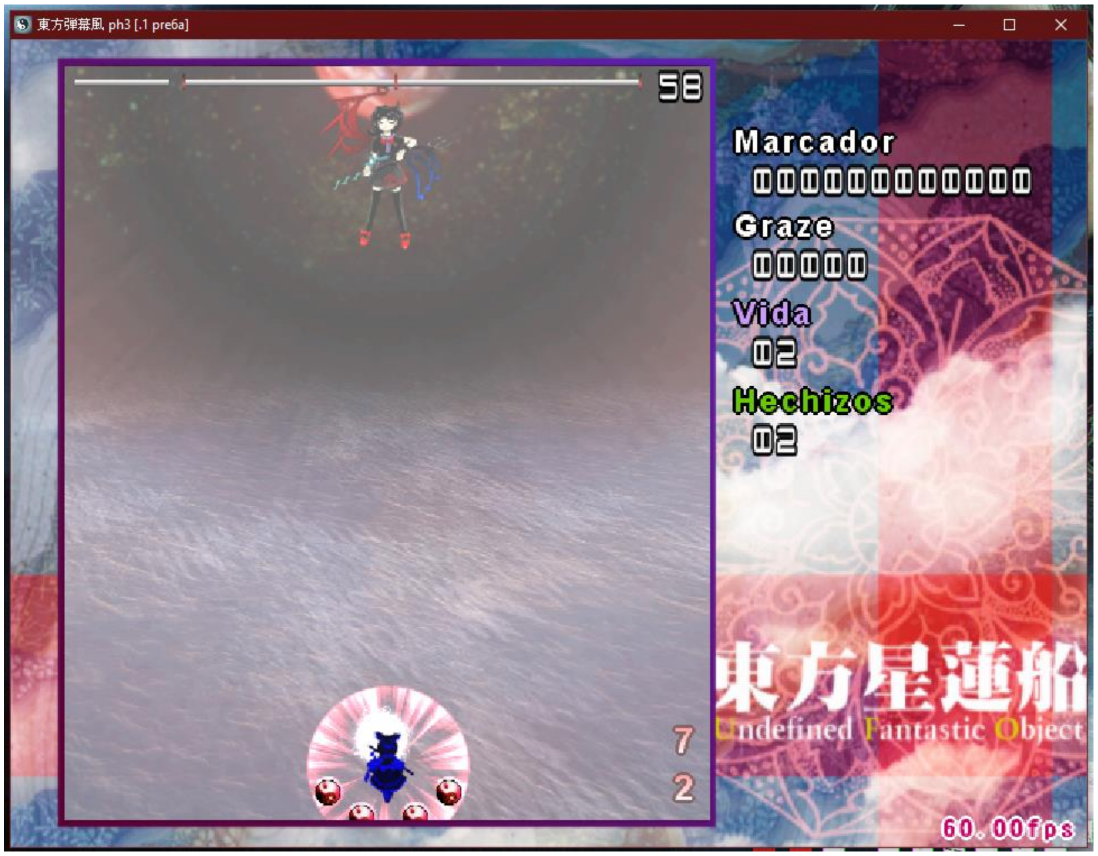


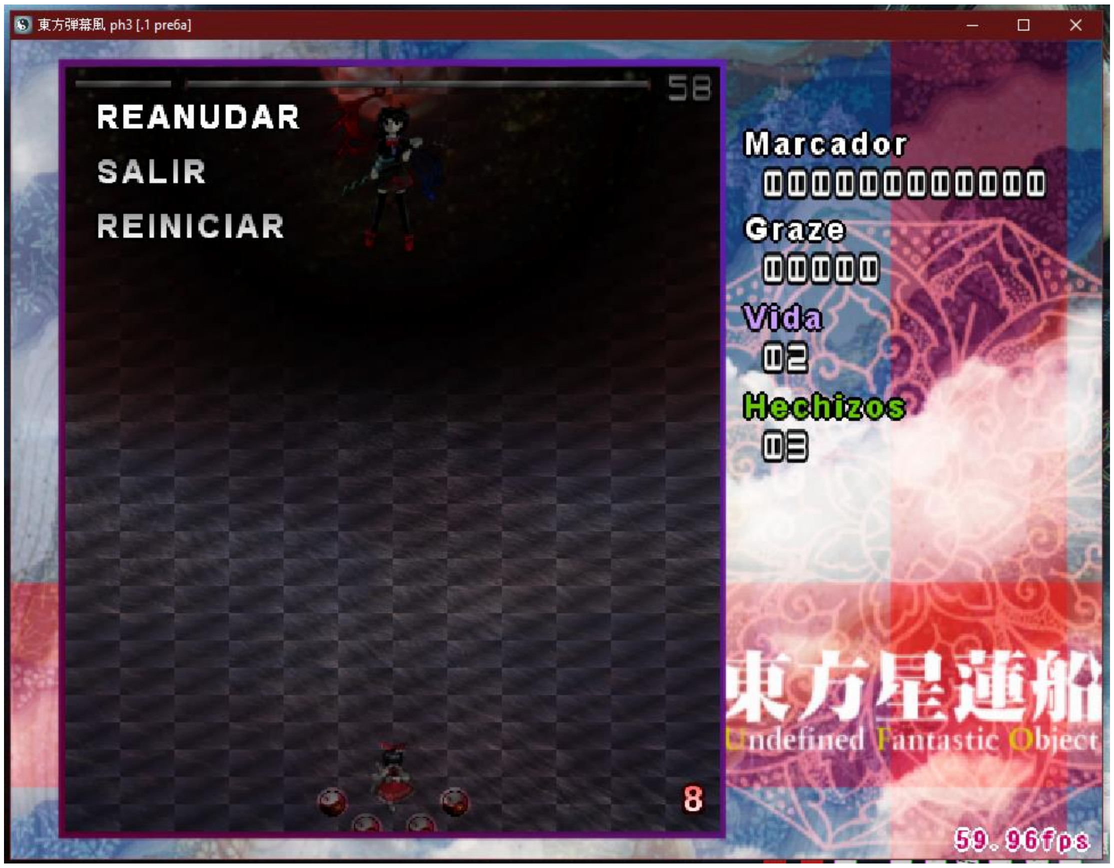

東方弾幕風 ph3 [.1 pre6a]
54
Marcador
0000000000000
Graze
00003
Vida
02
Hechizos
03
東方星蓮船
Undefined Fantastic Object
3
59.98fps

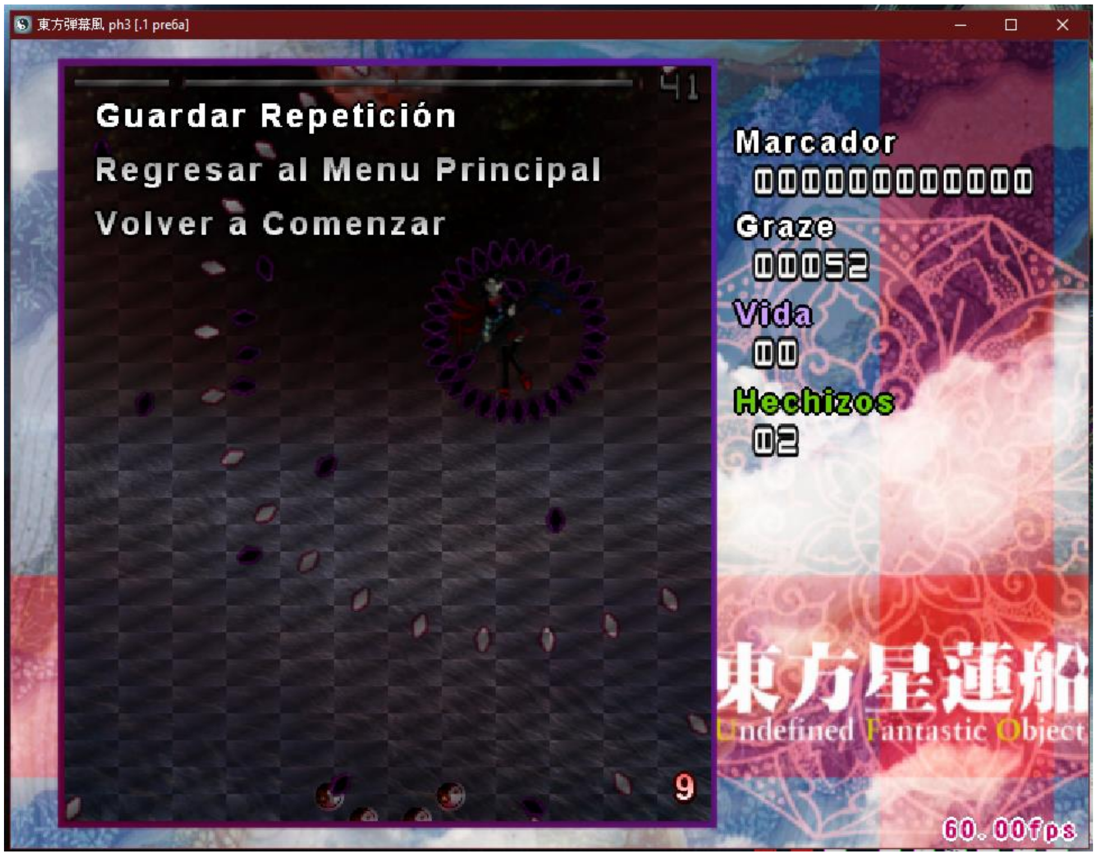

東方弾幕風 ph3 [.1 pre6a]
55
Surprising Rain "Guerrilla Typhoon"
Bonus 9258000 History 00/01
Marcador
0000000000000
Graze
00004
Vida
02
Hechizos
03
東方星蓮船
Undefined Fantastic Object
4
59.90fps


東方弾幕風 ph3 [.1 pre6a]
54
Marcador
0000000000000
Graze
00000
Vida
02
Hechizos
03
東方星蓮船
Undefined Fantastic Object
5
59.88fps

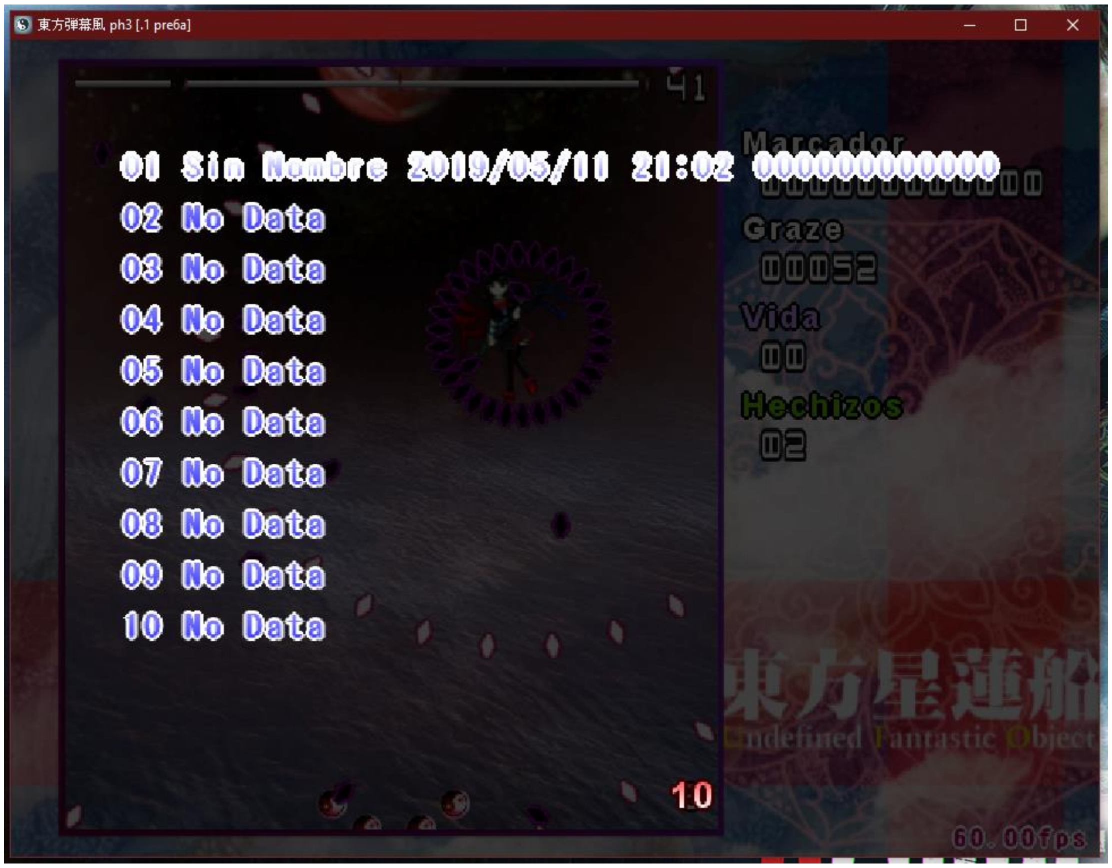

東方弾幕風 ph3 [.1 pre6a]
55
Ominous Clouds "Heian Dark Clouds"
Bonus 9194444 History 00/63
Marcador
0000000000000
Graze
00027
Vida
02
Hechizos
03
東方星蓮船
Undefined Fantastic Object
6
41.63fps


東方弾幕風 ph3 [.1 pre6a]
58
Marcador
0000000000000
Graze
00000
Vida
02
Hechizos
02
東方星蓮船
Undefined Fantastic Object
7
2
60.00fps


東方弾幕風 ph3 [.1 pre6a]
58
REANUDAR
SALIR
REINICIAR
Marcador
0000000000000
Graze
00000
Vida
02
Hechizos
03
東方星蓮船
Undefined Fantastic Object
8
59.96fps



東方弾幕風 ph3 [.1 pre6a]
Guardar Repetición
Regresar al Menu Principal
Volver a Comenzar
Marcador
0000000000000
Graze
00052
Vida
00
Hechizos
02
東方星蓮船
Undefined Fantastic Object
9
60.00fps



東方弾幕風 ph3 [.1 pre6a]
01 Sin Nombre 2019/05/11 21:02 000000000000
02 No Data
03 No Data
04 No Data
05 No Data
06 No Data
07 No Data
08 No Data
09 No Data
10 No Data
10
60.00fps

Repetir Selección

01 CHRIS 000000000000 DontWork 59.46fps 2019/05/11 21:56

CONCLUSIONES

Gracias a este proyecto pudimos entender mejor el funcionamiento de los autómatas dentro de aplicaciones (en este caso un videojuego), asi como su uso e importancia. Pudimos entender un poco mejor como describir el funcionamiento de un videojuego mediante una maquina de estados finitos.