//

// Created by Beniamin on 2017-05-17.

//

#include "Renderer.h"

#include <jni.h>

#include <android/log.h>

#include <string>

using namespace std;

extern "C" {

Renderer\* RENDERER;

Timer TIMER;

void Renderer::SetUp() {

square\_objects.current\_amount = 0;

for(int i = 0; i < MAX\_SQUARES\_AMOUNT; i++){

square\_objects.is\_taken[i] = false;

}

draw.draw = false;

draw.draw\_id = 0;

draw.position = 0;

draw.count = 0;

guianalog\_objects.created = 0;

players.amount = 0;

players.players = new Player[MAX\_PLAYERS];

SetPlatforms(20);

platforms.amount = 0;

}

int Renderer::CreateSquares(int amount, float \*\*dsxyz) {

int i = 0;

int count = 0;

while(count < amount && i < MAX\_SQUARES\_AMOUNT){

if(!square\_objects.is\_taken[i])

{

square\_objects.is\_taken[i] = true;

square\_objects.current\_amount++;

square\_objects.squares[i] = new Square(i);

square\_objects.squares[i]->Create(dsxyz[count][0], dsxyz[count][1], dsxyz[count][2], dsxyz[count][3]);

count++;

}

i++;

}

return square\_objects.current\_amount - 1;

}

void Renderer::LoadAnimations(int square\_id, int amount, float \*\*data) {

if(square\_id < MAX\_SQUARES\_AMOUNT){

if(square\_objects.is\_taken[square\_id]){

square\_objects.squares[square\_id]->LoadAnimations(amount, data);

}

}

}

bool Renderer::ToDraw(int drawed\_scene) {

if(draw.count < square\_objects.current\_amount){

for(int i = draw.position; i < MAX\_SQUARES\_AMOUNT; i++){

if(square\_objects.is\_taken[i]){

draw.count++;

if(square\_objects.squares[i]->Visible()

&& square\_objects.squares[i]->DrawedScene() == drawed\_scene){

//\_\_android\_log\_print(ANDROID\_LOG\_VERBOSE, "SCENE", to\_string(drawed\_scene).c\_str(), 1);

draw.draw\_id = i;

draw.draw = true;

draw.position = i+1;

return draw.draw;

}

}

}

draw.count = 0;

draw.position = 0;

draw.draw = false;

return draw.draw;

}else{

draw.count = 0;

draw.position = 0;

draw.draw = false;

return draw.draw;

}

}

void Renderer::ReadDrawedSquareRect(){

if(draw.draw)

square\_objects.squares[draw.draw\_id]->SquareRect(matrix.square\_vs);

}

void Renderer::ReadDrawedTextureRect(){

if(draw.draw)

square\_objects.squares[draw.draw\_id]->TextureRect(matrix.texture\_vs);

}

void Renderer::ScreenToRender(int drawed\_scene, float &x, float &y){

int l\_l;

int l\_r;

if(drawed\_scene == GAME\_SCENE){

l\_l = 1;

l\_r = -1;

}else {

l\_l = -1;

l\_r = 1;

}

float sourceTopLeftX = 0.0f;

float sourceTopLeftY = 0.0f;

float sourceBottomRightX = screen.screen\_w;

float sourceBottomRightY = screen.screen\_h;

float targetTopLeftX = l\_l \* (float)(screen.screen\_w/screen.screen\_h);

float targetTopLeftY = 1;

float targetBottomRightX = l\_r \* (float)(screen.screen\_w/screen.screen\_h);

float targetBottomRightY = -1;

x = targetTopLeftX +

((x - sourceTopLeftX) / (sourceBottomRightX-sourceTopLeftX))\*(targetBottomRightX-targetTopLeftX);

y = targetTopLeftY +

((y - sourceTopLeftY) / (sourceBottomRightY-sourceTopLeftY))\*(targetBottomRightY-targetTopLeftY);

}

void Renderer::UpdateSquares(){

int count = 0;

int i = 0;

int l\_square\_id;

//Update All Animations

while(count < square\_objects.current\_amount && i < MAX\_SQUARES\_AMOUNT){

if(square\_objects.is\_taken[i]){

if(square\_objects.squares[i]->Visible()){

square\_objects.squares[i]->Update();

}

count++;

}

i++;

}

//Update Position of Analog Squares

for(int i = 0; i < guianalog\_objects.created; i++){

l\_square\_id = guianalog\_objects.analogs[i].SquareId();

if(square\_objects.squares[l\_square\_id]->Visible()){

square\_objects.squares[l\_square\_id]->SetCoords(guianalog\_objects.analogs[i].RenderInsideX(),

guianalog\_objects.analogs[i].RenderInsideY(),

0.0f);

}

}

//Update position of player squares

for(int i = 0; i < players.amount; i++){

l\_square\_id = players.players[i].SquareID();

if(square\_objects.squares[l\_square\_id]->Visible()){

if(players.players[i].AnimationId() != square\_objects.squares[l\_square\_id]->AnimationId()){

square\_objects.squares[l\_square\_id]->Show(players.players[i].AnimationId());

}

square\_objects.squares[l\_square\_id]->SetSide(players.players[i].RightSide());

square\_objects.squares[l\_square\_id]->SetCoords(players.players[i].X(),

players.players[i].Y(),

players.players[i].Z());

}

}

FPS++;

if(timer.Current() >= 1){

\_\_android\_log\_print(ANDROID\_LOG\_VERBOSE, "FPS", to\_string(FPS).c\_str(), 1);

timer.Start();

FPS = 0;

}

}

void Renderer::CreateAnalogs(int amount, float \*\*data) {

guianalog\_objects.created = amount;

guianalog\_objects.analogs = new GUIAnalog[amount];

for(int i = 0; i < amount; i++){

guianalog\_objects.analogs[i].Create(i, data[i][0], data[i][1],

data[i][2], data[i][3], data[i][4], data[i][5]);

}

}

void Renderer::HandleAnalogs(){

int l\_pointer\_id;

float l\_render\_x;

float l\_render\_y;

bool l\_found = false;

for(int i = 0; i < guianalog\_objects.created; i++){

l\_found = false;

if(input.IsActive(0)){

l\_pointer\_id = 0;

l\_render\_x = input.PointerX(l\_pointer\_id);

l\_render\_y = input.PointerY(l\_pointer\_id);

ScreenToRender(FRONT\_GUI, l\_render\_x, l\_render\_y);

l\_found = guianalog\_objects.analogs[i].Check(l\_render\_x, l\_render\_y, l\_pointer\_id);

}

if(!l\_found && input.IsActive(1)){

l\_pointer\_id = 1;

l\_render\_x = input.PointerX(l\_pointer\_id);

l\_render\_y = input.PointerY(l\_pointer\_id);

ScreenToRender(FRONT\_GUI, l\_render\_x, l\_render\_y);

l\_found = guianalog\_objects.analogs[i].Check(l\_render\_x, l\_render\_y, l\_pointer\_id);

}

if(!l\_found) guianalog\_objects.analogs[i].Reset();

guianalog\_objects.analogs[i].SetInputState();

}

//0 IS LEFT AND 1 IS RIGHT

}

void Renderer::ProcessInput(float \*n\_data) {

input.ProcessInput(n\_data, screen.screen\_w, screen.screen\_h);

HandleAnalogs();

}

void Renderer::AddPlayer(player\_data l\_data, player\_propereties l\_prop, int l\_frames\_amount,

player\_frame \*l\_frames) {

if(players.amount < MAX\_PLAYERS){

l\_data.id = players.amount;

players.players[players.amount].Create(l\_data, l\_prop);

players.players[players.amount].AddFrames();

players.amount++;

}

}

void Renderer::HandleInput() {

if(players.amount > 0) {

input.SetPlayerLIS(guianalog\_objects.analogs[0].InputState(), guianalog\_objects.analogs[0].Hold(),

guianalog\_objects.analogs[1].InputState(), guianalog\_objects.analogs[1].Hold());

if(players.players[0].RequestInput()){

players.players[0].SetLogicInputState(input.PlayerLIS());

}

}

}

void Renderer::Logic(){

for(int i = 0; i < players.amount; i++){

players.players[i].Update();

}

for(int i = 0; i < players.amount; i++){

players.players[i].Move();

}

}

void Renderer::Physic() {

//Player Ground State

float \*l\_player\_last\_xyz = new float[3];

float \*l\_player\_current\_xyz = new float[3];

float \*l\_player\_size = new float[2];

for(int i = 0; i < players.amount; i++){

l\_player\_last\_xyz[0] = players.players[i].LastX();

l\_player\_last\_xyz[1] = players.players[i].LastY();

l\_player\_last\_xyz[2] = players.players[i].LastZ();

l\_player\_current\_xyz[0] = players.players[i].X();

l\_player\_current\_xyz[1] = players.players[i].Y();

l\_player\_current\_xyz[2] = players.players[i].Z();

l\_player\_size[0] = 0.2f;

l\_player\_size[1] = 0.2f;

switch (players.players[i].GroundState()){

case GS\_NONE:

players.players[i].SetGroundState(0, GS\_IN\_AIR, 0);

for(int j = 0; j < platforms.amount; j++){

if(platforms.platforms[j].Check(l\_player\_last\_xyz, l\_player\_current\_xyz, l\_player\_size))

{

players.players[i].SetGroundState(j, GS\_ON\_PLATFORM, platforms.platforms[j].Y());

j = platforms.amount;

}

}

//check if he stands at any platform

break;

case GS\_IN\_AIR:

for(int j = 0; j < platforms.amount; j++){

if(platforms.platforms[j].Check(l\_player\_last\_xyz, l\_player\_current\_xyz, l\_player\_size))

{

players.players[i].SetGroundState(j, GS\_ON\_PLATFORM, platforms.platforms[j].Y());

j = platforms.amount;

}

}

break;

case GS\_ON\_PLATFORM:

if(!platforms.platforms[players.players[i].PlatformId()].Check(l\_player\_last\_xyz, l\_player\_current\_xyz, l\_player\_size))

{

players.players[i].SetGroundState(0, GS\_IN\_AIR, 0);

}

break;

default:

//something is wrong ground platform should be only one of above

break;

}

}

for(int i = 0; i < players.amount; i++){

players.players[i].Gravity( 2, 1);

players.players[i].Friction(0.1f, 6.0f);

}

//CLEANUP

delete [] l\_player\_last\_xyz;

delete [] l\_player\_current\_xyz;

delete [] l\_player\_size;

}

void Renderer::AddPlatform(int square\_id, int type, float \*rect) {

if(platforms.amount < platforms.max){

platforms.platforms[platforms.amount].Create(platforms.amount, square\_id, type, rect);

platforms.amount++;

}

};

void Renderer::LoadPlayerFrames(player\_frame\* n\_p\_frames, int \*n\_frames, int n\_amount) {

for(int i = 0; i < n\_amount; i++){

//MAMY 10 FALL

/\*n\_p\_frames[i].propereties.movement\_state = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+0];

n\_p\_frames[i].propereties.animation = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+1];

n\_p\_frames[i].no\_input.delta = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+2] / 1000.0f;

n\_p\_frames[i].no\_input.on\_ground\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+3];

n\_p\_frames[i].no\_input.in\_air\_frame\_id= n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+4];

n\_p\_frames[i].left.on\_ground\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+5];

n\_p\_frames[i].left.in\_air\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+6];

n\_p\_frames[i].right.on\_ground\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+7];

n\_p\_frames[i].right.in\_air\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+8];

n\_p\_frames[i].up\_left.on\_ground\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+9];

n\_p\_frames[i].up\_left.in\_air\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+10];

n\_p\_frames[i].up\_right.on\_ground\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+11];

n\_p\_frames[i].up\_right.in\_air\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+12];

n\_p\_frames[i].down\_left.on\_ground\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+13];

n\_p\_frames[i].down\_left.in\_air\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+14];

n\_p\_frames[i].down\_right.on\_ground\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+15];

n\_p\_frames[i].down\_right.in\_air\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+16];

n\_p\_frames[i].up.on\_ground\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+17];

n\_p\_frames[i].up.in\_air\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+18];

n\_p\_frames[i].down.on\_ground\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+19];

n\_p\_frames[i].down.in\_air\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+20];

n\_p\_frames[i].push\_a.on\_ground\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+21];

n\_p\_frames[i].push\_a.in\_air\_frame\_id = n\_frames[(i\*PLAYER\_FRAME\_DATA\_AMOUNT)+22];\*/

}

}

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

JNIEXPORT void JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_UpdateSquares(JNIEnv \*env, jobject instance) {

RENDERER->UpdateSquares();

}

JNIEXPORT jfloatArray JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_DrawSquareCoords(JNIEnv \*env, jobject instance) {

jfloatArray result;

jfloat \*fill = new float[12];

result = (\*env).NewFloatArray(12);

RENDERER->ReadDrawedSquareRect();

for(int i = 0; i < 12; i++){

fill[i] = RENDERER->DrawedSquareRect(i);

}

(\*env).SetFloatArrayRegion(result, 0, 12, fill);

delete [] fill;

return result;

}

JNIEXPORT jfloatArray JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_DrawTextureCoords(JNIEnv \*env, jobject instance) {

jfloatArray result;

jfloat \*fill = new float[8];

result = (\*env).NewFloatArray(8);

RENDERER->ReadDrawedTextureRect();

for(int i = 0; i < 8; i++){

fill[i] = RENDERER->DrawedTextureRect(i);

}

(\*env).SetFloatArrayRegion(result, 0, 8, fill);

delete [] fill;

return result;

}

JNIEXPORT jshortArray JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_DrawOrder(JNIEnv \*env, jobject instance) {

jshortArray result;

jshort \*fill = new jshort[6];

result = (\*env).NewShortArray(6);

for(int i = 0; i < 6; i++){

fill[i] = DRAW\_ORDER[i];

}

(\*env).SetShortArrayRegion(result, 0, 6, fill);

delete [] fill;

return result;

}

JNIEXPORT void JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_InitRender(JNIEnv \*env, jobject instance) {

RENDERER = new Renderer;

RENDERER->SetUp();

RENDERER->StartFps();

}

JNIEXPORT void JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_DeleteRender(JNIEnv \*env, jobject instance) {

delete RENDERER;

}

JNIEXPORT jint JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_AddSquares(JNIEnv \*env, jobject instance,

jint amount, jfloatArray data\_) {

jfloat \*data = env->GetFloatArrayElements(data\_, NULL);

float \*\*l\_data = new float\*[amount]; // there is 11 data elements to pass ATM

int result;

for(int i = 0; i < amount; i++){

l\_data[i] = new float[3];

for(int j = 0; j < 3; j++){

l\_data[i][j] = data[j];

}

}

result = RENDERER->CreateSquares(amount, l\_data);

delete [] l\_data;

env->ReleaseFloatArrayElements(data\_, data, 0);

return result;

}

JNIEXPORT void JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_SetScreen(JNIEnv \*env, jobject instance, jfloat w,

jfloat h) {

RENDERER->SetScreen(w, h);

}

JNIEXPORT jboolean JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_ToDraw(JNIEnv \*env, jobject instance,

jint drawed\_scene) {

return RENDERER->ToDraw(drawed\_scene);

}

JNIEXPORT void JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_LoadSquareAnimations(JNIEnv \*env,

jobject instance,

jint id,

jint amount,

jfloatArray data\_) {

jfloat \*data = env->GetFloatArrayElements(data\_, NULL);

float\*\* l\_data = new float\*[amount];

for(int i = 0; i < amount; i++){

l\_data[i] = new float[12];

l\_data[i][0] = data[(i\*12)+0];

l\_data[i][1] = data[(i\*12)+1];

l\_data[i][2] = data[(i\*12)+2];

l\_data[i][3] = data[(i\*12)+3];

l\_data[i][4] = data[(i\*12)+4];

l\_data[i][5] = data[(i\*12)+5];

l\_data[i][6] = data[(i\*12)+6];

l\_data[i][7] = data[(i\*12)+7];

l\_data[i][8] = data[(i\*12)+8];

l\_data[i][9] = data[(i\*12)+9];

l\_data[i][10] = data[(i\*12)+10];

l\_data[i][11] = data[(i\*12)+11];

}

RENDERER->LoadAnimations(id, amount, l\_data);

delete [] l\_data;

env->ReleaseFloatArrayElements(data\_, data, 0);

}

JNIEXPORT void JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_ProcessInput(JNIEnv \*env, jobject instance,

jfloatArray data\_) {

jfloat \*data = env->GetFloatArrayElements(data\_, NULL);

float \*l\_data = new float[5];

for(int i = 0; i < 5; i++){

l\_data[i] = data[i];

}

RENDERER->ProcessInput(data);

delete [] l\_data;

env->ReleaseFloatArrayElements(data\_, data, 0);

}

JNIEXPORT jboolean JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_AddThatSquare(JNIEnv \*env, jobject instance) {

return RENDERER->AddThatSquare();

}

JNIEXPORT jfloatArray JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_NewBorn(JNIEnv \*env, jobject instance) {

jfloatArray result;

jfloat \*fill = new float[2];

result = (\*env).NewFloatArray(2);

fill[0] = RENDERER->InputX();

fill[1] = RENDERER->InputY();

RENDERER->ScreenToRender(GAME\_SCENE, fill[0], fill[1]);

(\*env).SetFloatArrayRegion(result, 0, 2, fill);

delete [] fill;

return result;

}

JNIEXPORT void JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_CreateAnalogs(JNIEnv \*env, jobject instance,

jint amount, jfloatArray data\_) {

jfloat \*data = env->GetFloatArrayElements(data\_, NULL);

float\*\* l\_data = new float\*[amount];

for(int i = 0; i < amount; i++){

l\_data[i] = new float[6];

for(int j = 0; j < 6; j++){

l\_data[i][j] = data[(i\*6)+j];

}

}

RENDERER->CreateAnalogs(amount, l\_data);

env->ReleaseFloatArrayElements(data\_, data, 0);

delete [] l\_data;

}

JNIEXPORT void JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_HandleInput\_\_(JNIEnv \*env, jobject instance) {

RENDERER->HandleInput();

}

JNIEXPORT void JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_CreatePlayer(JNIEnv \*env, jobject instance,

jfloatArray n\_player\_data\_,

jint n\_frames\_amount,

jintArray n\_frames\_) {

jfloat \*n\_player\_data = env->GetFloatArrayElements(n\_player\_data\_, NULL);

jint \*n\_frames = env->GetIntArrayElements(n\_frames\_, NULL);

player\_data l\_pdata;

player\_propereties l\_pprop;

player\_frame\* l\_pframes;

l\_pframes = new player\_frame[n\_frames\_amount];

//CREATE PLAYER DATA

l\_pdata.id = 0; //in a table

l\_pdata.x = n\_player\_data[5];

l\_pdata.y = n\_player\_data[6];

l\_pdata.z = n\_player\_data[7];

l\_pdata.last\_x = n\_player\_data[5];

l\_pdata.last\_y = n\_player\_data[6];

l\_pdata.last\_z = n\_player\_data[7];

l\_pdata.suqare\_id = n\_player\_data[8];

l\_pdata.air\_jump = 0;

l\_pdata.ground\_state = 0;

l\_pdata.movement\_state = 0;

l\_pdata.animation = 0;

//CREATE PLAYER PROP

l\_pprop.movement\_velocity = n\_player\_data[0];

l\_pprop.jump\_power = n\_player\_data[1];

l\_pprop.air\_jump\_amount = n\_player\_data[2];

l\_pprop.air\_vertical\_velocioty = n\_player\_data[3];

l\_pprop.fall\_velocity = n\_player\_data[4];

//LOAD PLAYER FRAMES

RENDERER->LoadPlayerFrames(l\_pframes, n\_frames, n\_frames\_amount);

RENDERER->AddPlayer(l\_pdata, l\_pprop, n\_frames\_amount, l\_pframes);

delete [] l\_pframes;

env->ReleaseFloatArrayElements(n\_player\_data\_, n\_player\_data, 0);

env->ReleaseIntArrayElements(n\_frames\_, n\_frames, 0);

}

JNIEXPORT void JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_Logic(JNIEnv \*env, jobject instance) {

RENDERER->Logic();

}

JNIEXPORT void JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_CreatePlatform(JNIEnv \*env, jobject instance,

jint square\_id, jint type,

jfloatArray rect\_) {

jfloat \*rect = env->GetFloatArrayElements(rect\_, NULL);

float\* l\_rect = new float[5];

for(int i = 0; i < 5; i++){

l\_rect[i] = rect[i];

}

RENDERER->AddPlatform(square\_id, type, l\_rect);

// TODO

env->ReleaseFloatArrayElements(rect\_, rect, 0);

}

JNIEXPORT void JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_Physic(JNIEnv \*env, jobject instance) {

// TODO

RENDERER->Physic();

}

JNIEXPORT void JNICALL

Java\_com\_natywna\_aplikacja\_cplusplus\_MyGLRenderer\_Pause(JNIEnv \*env, jobject instance, jint mss) {

//float l\_delta;

TIMER.Start();

while(TIMER.Current() < mss){

// l\_delta = TIMER.Current();

//\_\_android\_log\_print(ANDROID\_LOG\_VERBOSE, "DELTA", to\_string(l\_delta).c\_str(), 1);

}

TIMER.Stop();

}

}