Họ và tên : Nguyễn Bá Quí

MSSV : 1311252

BÁO CÁO CUỐI KÌ ANDROID

**Đề tài :** Xây dựng game “Tìm điểm khác nhau giữa hai ảnh”

**1. Giới thiệu**

Xây dựng game với giao diện đơn giản , có đầy đủ các tính năng như các game cùng loại khác:

- Hỗ trợ Zoom và scroll ảnh, áp dụng đồng thời các chuyển động lên cả 2 ảnh original và mirror

- Đánh dấu các vùng khác nhau khi người chơi chạm vào điểm khác biệt

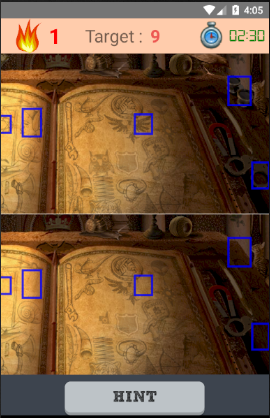
D:\spot-differences\app\src\main\res\drawable\target.png- Chớp dấu  khi người chới chạm sai

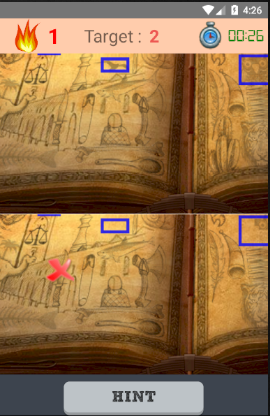
- Có nút hint, load trong một khoảng thời gian, zoom và target vào điểm khác biệt

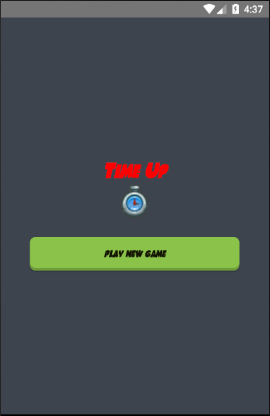
- Giới hạn thời gian chơi cho mỗi mission

- Độ khó mission tăng dần

Một số hình ảnh :







**2. Các nguồn ngoài**

**-** <https://github.com/MikeOrtiz/TouchImageView>

Widget hỗ trợ đầy đủ khả năng Zoom ảnh và scroll như trong thư viện ảnh của Google : Animation DoubleTapZoom, ScaleGestureDetector, GestureDetector .

Để 2 ảnh thực hiện đồng thời các chuyển động :

**topImage**.setOnTouchImageViewListener(**new** TouchImageView.OnTouchImageViewListener() {

@Override  
 **public void** before() { . . . }  
 @Override  
 **public void** onMove() {  
 **bottomImage**.setZoom(**topImage**);  
 }  
});  
**bottomImage**.setOnTouchImageViewListener(**new** TouchImageView.OnTouchImageViewListener() {

@Override  
 **public void** before() { . . . }  
 @Override  
 **public void** onMove() {  
 **topImage**.setZoom(**bottomImage**);  
 }  
});

onMove() được gọi cuối cùng trong PrivateOnTouchListener **implements** OnTouchListener của TouchImageView

**mScaleDetector** = **new** ScaleGestureDetector(context, **new** ScaleListener());  
**mGestureDetector** = **new** GestureDetector(context, **new** GestureListener());

**super**.setOnTouchListener(**new** PrivateOnTouchListener());

**private class** PrivateOnTouchListener **implements** OnTouchListener {  *// Remember last point position for dragging* **private** PointF **last** = **new** PointF();  
 @Override  
 **public boolean** onTouch(View v, MotionEvent event) {

*// before action*

**if** (**touchImageViewListener** != **null**)  
 **touchImageViewListener**.before();  
 **mScaleDetector**.onTouchEvent(event);  
 **mGestureDetector**.onTouchEvent(event);  
 PointF curr = **new** PointF(event.getX(), event.getY());

[BODY …] *// SET STATE AND MATRIX*

*// OnTouchImageViewListener is set: TouchImageView dragged by user* **if** (**touchImageViewListener** != **null**) {  
 **touchImageViewListener**.onMove();  
 } *// indicate event was handled* **return true**;  
 }  
}

onMove() cũng được gọi trong hàm run() của DoubleTapZoom **implements** Runnable

Code animation zoom vào một tọa độ trên layout (áp dụng đối với cả tọa độ âm) trên View:

DoubleTapZoom doubleTap = **new** DoubleTapZoom(targetZoom, **touch.x**, **touch.y**, **true**);  
compatPostOnAnimation(doubleTap);

- <https://gist.github.com/bverc/1492672>

Một CountDownTimer thay thế cho built-in CountDownTimer của Android, hỗ trợ thêm pause() và resume(). Có 2 CountDownTimer sử dụng trong project này là mission Timer và hint loading Timer

- <https://github.com/hoang8f/android-flat-button>

Thiết kế flat button đơn giản

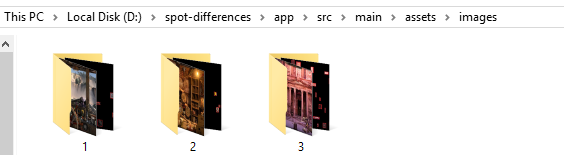
- Addimplementation **'com.googlecode.json-simple:json-simple:1.1'** vào dependencies

Hồ trợ đọc file JSON dạng JSONArray , dùng để load database

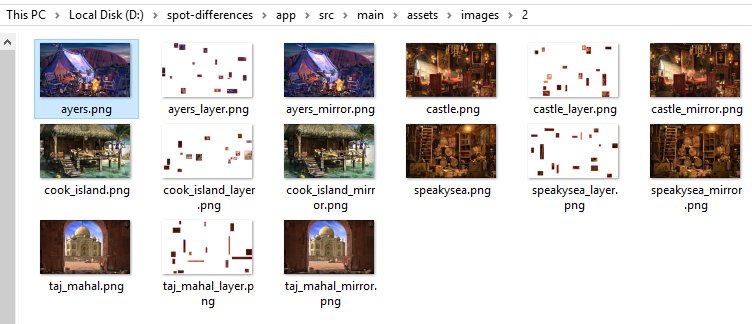
**3. Tài nguyên**

Nguồn ảnh được lấy từ game Amazing Adventure – Around the world

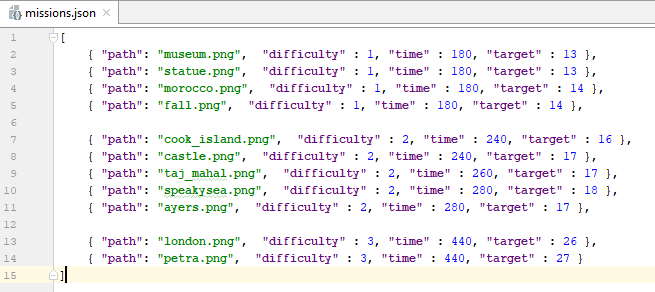
Ảnh được đặt trong thư mục assets/images tương ứng với 3 cấp độ



Mỗi mission ngoài 2 ảnh original và mirror để so sánh, còn có một ảnh gọi là “layer” lưu giữ vị trí điểm khác nhau



Sử dụng file missions.json để lưu thông tin các màn chơi



Ta chỉ cần original path, mirror path và layer path được suy ra dễ dàng

**public class** SingleMission {**public** String **path**;  
 **public int difficulty**;  
 **public int time**;  
 **public int target**; *// number of ares need to be spotted  
 // unique color used to check if block was marked, it mean very almost TRANSPARENT* **public static final int *CHECKED\_BLOCK*** = 0x01000000;

**public** SingleMission(){  
 **path** = **""**;  
 }  
  
 **public** SingleMission(String path, **int** difficulty, **int** time, **int** target){  
 **this**.**path** = path;  
 **this**.**difficulty** = difficulty;  
 **this**.**time** = time;  
 **this**.**target** = target;  
 }  
  
 **public** String getPathInAssets(){  
 **return "images/"**+**difficulty**+**"/"**+ **path**;  
 }  
  
 **public** String getMirrorPathInAssets(){  
 String name = **path**.split(**"\\."**)[0];  
 **return "images/"** + **difficulty** + **"/"** + name + **"\_mirror.png"**;  
 }  
  
 **public** String getLayerPathInAssets() {  
 String name = **path**.split(**"\\."**)[0];  
 **return "images/"** + **difficulty** + **"/"** + name + **"\_layer.png"**;

}

. . . .

}

Đọc file json và lưu các đối tượng vào các **static** ArrayList<SingleMission>

**public class** MissionDatabase {  
 **public static** ArrayList<SingleMission> *EasyMissions* = **new** ArrayList<>();  
 **public static** ArrayList<SingleMission> *MediumMissions* = **new** ArrayList<>();  
 **public static** ArrayList<SingleMission> *HardMissions* = **new** ArrayList<>();

*// it will generate static ArrayLists of database from JSON Source* **public static void** getMissionsFromFile(String filename, Context context){

**try** {  
 *// Load file* JSONParser parser = **new** JSONParser();  
 String jsonString = AssetReader.*loadJsonFromAsset*(filename, context);  
 JSONArray missions = (JSONArray) parser.parse(jsonString);

*// Get SingleMission objects from data* **for**(Object o : missions){  
 JSONObject jo = (JSONObject) o;  
 SingleMission mission = **new** SingleMission();  
  
 mission.**path** = (String) jo.get(**"path"**) ;  
 mission.**difficulty** = ((Long) jo.get(**"difficulty"**)).intValue();  
 mission.**time** = ((Long) jo.get(**"time"**)).intValue();  
 mission.**target** = ((Long) jo.get(**"target"**)).intValue();  
  
 **if** (mission.**difficulty** == 1)  
 *EasyMissions*.add(mission);  
 **else if** (mission.**difficulty** == 2)  
 *MediumMissions*.add(mission);  
 **else** *HardMissions*.add(mission);  
 }  
 } **catch** (ParseException e) { e.printStackTrace(); }  
 }

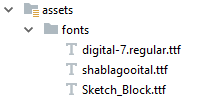
**public static** SingleMission get(**int** index, **int** difficulty){  
 **if** (difficulty == 1)  
 **return** *EasyMissions*.get(index);  
 **else if** (difficulty == 2)  
 **return** *MediumMissions*.get(index);  
 **else  
 return** *HardMissions*.get(index);  
 }

}

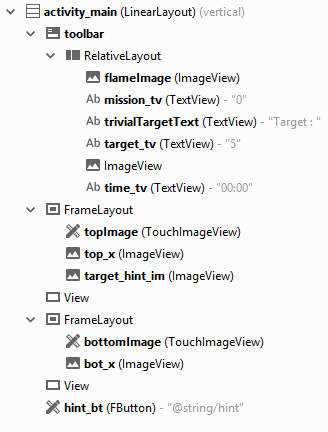
Trong MainGameActivity, để khởi tạo các mission một cách ngẫu nhiên

MissionDatabase.*getMissionsFromFile*(**"missions.json"**, **this**);  
  
*//Now we gonna shuffle the elements of the list so that we will get mission at specific difficulty randomly*Collections.*shuffle*(MissionDatabase.*EasyMissions*);  
Collections.*shuffle*(MissionDatabase.*MediumMissions*);  
Collections.*shuffle*(MissionDatabase.*HardMissions*);  
*//set first mission and states*updateQueue();

Add một số fonts sử dụng trong project



**4. Main Layout**



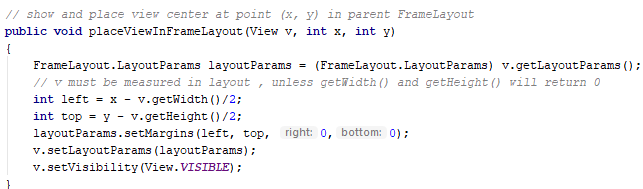


Sử dụng FrameLayout, ta có thể chồng các View nhỏ trên TouchImageView

Các mark ban đầu được set INVISIBLE

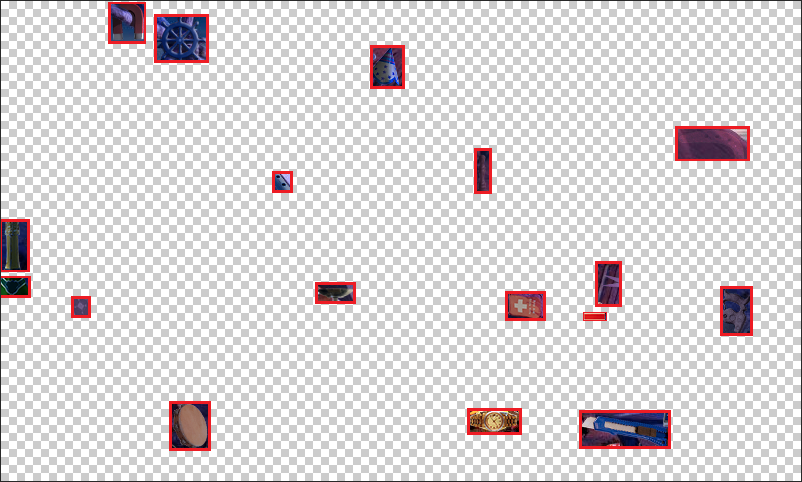
**top\_wrong\_tick**.setVisibility(ImageView.***INVISIBLE***);  
**bottom\_wrong\_tick**.setVisibility(ImageView.***INVISIBLE***);  
**target\_hint\_im**.setVisibility(View.***INVISIBLE***);

Để hiện view tại một ví trí trong frame layout :



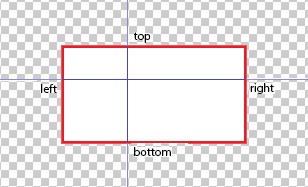
**5. Xử lý singleTap on Image:**

Trước hết, ta nói sơ về ảnh layer lưu giữ vị trí các vị trí khác biệt giữa 2 ảnh original và mirror



Các điểm khác nhau được biểu diễn bởi các block (Rect) và rời nhau. Phần còn lại là Transparent background

Từ một điểm thuộc block, ta có thể suy ra dễ dàng các tọa độ của block



Có tất cả 3 trạnh thái của pixel trên layer bitmap :

* pixel == Color.TRANSPARENT

pixel không thuộc bất cứ block nào, khi player kích vào sẽ chớp dấu 

* pixel != Color.TRANSPARENT và pixel != SingleMission.CHECKED\_BLOCK

pixel thuộc trong một các block chưa được dò

Khi player kích vào ta sẽ lấy Rect chứa pixel đó và vẽ Rect lên canvas của 2 ảnh

Đồng thời, set các giá trị pixel trong block bằng một giá trị đặc biệt CHECKED\_BLOCK trên layer bitmap

* pixel == SingleMission.CHECKED\_BLOCK

Bỏ qua, block không được dò lại, không có điều gì xảy ra khi player kích vào

Add singleTap Event

**topImage**.setOnDoubleTapListener(**new** GestureDetector.OnDoubleTapListener() {  
 @Override  
 **public boolean** onSingleTapConfirmed(MotionEvent e) {  
 **return** singleTapImageAction(e, **top\_wrong\_tick**);  
 }  
 @Override  
 **public boolean** onDoubleTap(MotionEvent e) { **return false**; }  
 @Override  
 **public boolean** onDoubleTapEvent(MotionEvent e) { **return false**; }  
});  
  
**bottomImage**.setOnDoubleTapListener(**new** GestureDetector.OnDoubleTapListener() {  
 @Override  
 **public boolean** onSingleTapConfirmed(MotionEvent e) {  
 **return** singleTapImageAction(e, **bottom\_wrong\_tick**);  
 }  
 @Override  
 **public boolean** onDoubleTap(MotionEvent e) { **return false**; }  
 @Override  
 **public boolean** onDoubleTapEvent(MotionEvent e) { **return false**; }  
  
});

Sử dụng method transformCoordTouchToBitmap trong TouchImageView để chuyển tọa độ trong layout thành tọa độ của pixel trong bitmap

**public boolean** singleTapImageAction(MotionEvent e, ImageView wrong\_tick){  
 PointF pixel = **topImage**.transformCoordTouchToBitmap(e.getX(), e.getY(), **true**);  
 *// check wrong touch* **if** (**layerBitmap**.getPixel((**int**) pixel.**x**, (**int**)pixel.**y**) == Color.***TRANSPARENT***){  
 flashWrongTick(wrong\_tick, (**int**) e.getX(), (**int**)e.getY(), 200);  
 **return true**;  
 }

Rect block = SingleMission.*getBlock*(pixel.**x**, pixel.**y**, **layerBitmap**);

**if** (block == **null**) *// pixel was checked*  
 **return false**;  
  
 *// draw canvas  
 // it doesn't result in bitmap, just paint over image via canvas layer*

**topCanvas**.drawRect(block, **paint**);  
 **bottomCanvas**.drawRect(block, **paint**);  
 *// update view* **topImage**.setImageBitmap(**topBitmap**);  
 **bottomImage**.setImageBitmap(**bottomBitmap**);

**target** -= 1;  
 *// mark checked block by changing pixel to a unique value* **for** (**int** x = block.**left**; x <= block.**right**; x++)  
 **for** (**int** y = block.**top**; y <= block.**bottom**; y++)  
 **layerBitmap**.setPixel(x, y,SingleMission.***CHECKED\_BLOCK***);  
  
 **target\_tv**.setText(String.*valueOf*(**target**));

**if** (**target** == 0){  
 *// HANDLE WHEN MISSION END* disableImage();  
 **if** (**level** == **NUMBER\_EASY\_GAMES** + **NUMBER\_MEDIUM\_GAMES** + **NUMBER\_HARD\_GAMES**)  
 gameWon();  
 **else** nextMissionDialog();  
 }  
 **return true**;  
}

Xử lý flashWrongTick , ta chỉ đơn giản set View Visible trong một khoảng thời gian

**public void** flashWrongTick(**final** ImageView wrong\_tick, **int** x, **int** y, **int** duration){  
 *// place tick at touch point* placeViewInFrameLayout(wrong\_tick, x, y);  
 Handler handler = **new** Handler();  
 handler.postDelayed(**new** Runnable() {  
 @Override  
 **public void** run() {  
 wrong\_tick.setVisibility(ImageView.***GONE***);  
 }  
 }, duration);  
}

**6. Xử lý khi mission kết thúc**

Khi player tìm được điểm khác nhau cuối cùng (target giảm về 0 trong singleTapImageAction())

Có 2 trường hợp :

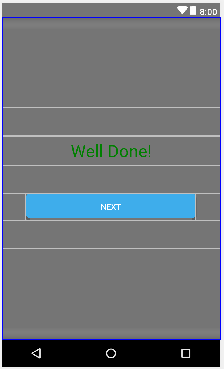
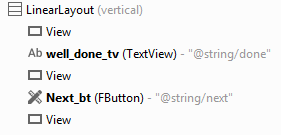
- Đây là mission cuối cùng (player có thể giới hạn số mission trong mỗi lần chơi), chuyển hướng về màn hình chúc mừng

**public void** gameWon() {  
 Intent intent = **new** Intent(**this**, GameWon.**class**);  
 startActivity(intent);  
 finish();  
}

- Chưa phải mission cuối cùng, hiển thị Dialog để dẫn đến màn tiếp theo

gọi onPause() và chờ người chơi nhấn vào Next

*//This will pause the timer*@Override  
**protected void** onPause() {  
 **super**.onPause();  
 **missionTimer**.pause();  
 **if** (**hintTimer** != **null**)  
 **hintTimer**.pause();  
}



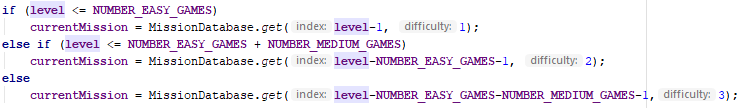
Background : #88000000

**public void** nextMissionDialog() {  
 **final** Dialog nextMissionDialog = **new** Dialog(MainGameActivity.**this**);  
 nextMissionDialog.requestWindowFeature(Window.***FEATURE\_NO\_TITLE***);  
 **if** (nextMissionDialog.getWindow() != **null**) {  
 ColorDrawable colorDrawable = **new** ColorDrawable(Color.***TRANSPARENT***);  
 nextMissionDialog.getWindow().setBackgroundDrawable(colorDrawable);  
 }  
 nextMissionDialog.setContentView(R.layout.***dialog\_next\_mission***);  
 nextMissionDialog.setCancelable(**false**);  
 nextMissionDialog.show();  
 *//Since the dialog is show to user just pause the timer in background* onPause();  
  
 TextView doneText = (TextView) nextMissionDialog.findViewById(R.id.***well\_done\_tv***);  
 FButton buttonNext = (FButton) nextMissionDialog.findViewById(R.id.***Next\_bt***);  
 Typeface sb = Typeface.*createFromAsset*(getAssets(), **"fonts/shablagooital.ttf"**);  
 doneText.setTypeface(sb);  
 buttonNext.setTypeface(sb);  
  
 *//OnCLick listener to go next mission* buttonNext.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 nextMissionDialog.dismiss(); *// dismiss dialog  
 // set up next level* **level**++ ;  
 updateQueue();  
 enableImage();  
 **if** (**hintTimer** != **null**)  
 **hintTimer**.resume();  
 }  
 });  
}



**7. Khởi tạo mission :** (hàm updateQueue)

Đầu tiên lấy đối tượng SingleMission theo level hiện tại



Lấy các giá trị và setText cho TextViews

**target** = **currentMission**.**target**;  
**timeValue** = **currentMission**.**time**;  
**mission\_tv**.setText(String.*valueOf*(**level**));  
**target\_tv**.setText(String.*valueOf*(**target**));

Chạy timer cho mission.

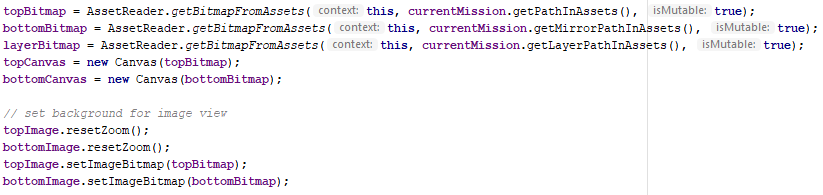
Mỗi lần tick, update time TextView , đồng thời giảm thời gian xuống 1 giây

Nếu player không tìm hết tất cả các điểm khác biệt trong thời gian đặt ra, hàm onFinish()

sẽ được gọi và dẫn tới màn hình Time Up

*//just set timer for next mission***missionTimer** = **new** CountDownTimer(**timeValue** \* 1000, 1000) {  
 **public void** onTick(**long** millisUntilFinished) {  
 **time\_tv**.setText(mls2str(**timeValue**));  
 *//With each iteration decrement the time by 1 sec* **timeValue** -= 1;  
 }  
  
 *//Now user is out of time* **public void** onFinish() {  
 *// navigate him to the time up activity* Intent intent = **new** Intent(**this**, Time\_Up.**class**);  
 startActivity(intent);  
 finish();

}  
}.start();

Cuối cùng, lấy bitmap và canvas ứng với các ảnh

**8. Tính năng Hint :**

Mục tiêu là tìm một block chưa được dò, ta chỉ cần tìm một pixel không phải là TRANSPARENT hoặc CHECKED\_BLOCK

Để search Pixel một cách ngẫu nhiên, có 2 hướng thường được dùng

- Search theo hướng ngẫu nhiên, ví dụ :

+ search from left to right, top to bottom

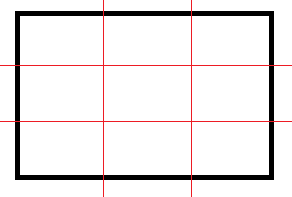
+ search from right to left, bottom to top

+ search from top to bottom, right to left

+ search from bottom to top, left to right

+ search from center

- Search theo vùng



Chia bitmap thành các vùng khác nhau, sau đó search theo thứ tự ngẫu nhiên các vùng

Dĩ nhiên , ta có thể kết hợp cả 2 phương pháp để ra phương pháp tốt nhất

**hintButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 directToHint();  
 hintLoading(10\*1000,100);  
 }  
});

* Hàm directToHint

Đầu tiên tìm kiếm một block, sau đó lấy điểm chính giữa của block để focus vào

Rect r = SingleMission.*hint*(**layerBitmap**);  
**if** (r == **null**)  
 **return**;

PointF pixelFocus = **new** PointF();  
pixelFocus.set((r.**left** + r.**right**)/2, (r.**top**+ r.**bottom**)/2);  
**topImage**.performZoomAtPixelWithAnimation(pixelFocus.**x**, pixelFocus.**y**);  
*// bottomImage.setZoom(topImage); // unneccesary, onMove() did it*

Hàm performZoomAtPixelWithAnimation chỉ đơn giản chuyển pixel thành tọa độ trên layout, sau đó mô phỏng doubleTapZoom tại tọa độ đó (có thể âm)

**public void** performZoomAtPixelWithAnimation(**float** x, **float** y){  
 PointF touch = transformCoordBitmapToTouch(x, y);  
 DoubleTapZoom doubleTap = **new** DoubleTapZoom(**maxScale**, touch.**x**, touch.**y**, **false**);  
 compatPostOnAnimation(doubleTap);

Tuy nhiên trong quá trình run chạy song song và không cập nhật ngay lập tức các giá trị của matrix dùng cho code sau đó

Vì thế, để tính toán ngay lập tức các giá trị của matrix, ta cần sử dụng setZoom

*// update MTRANS\_X and MTRANS\_Y immediately* Drawable drawable = getDrawable();  
 **int** w = drawable.getIntrinsicWidth();  
 **int** h = drawable.getIntrinsicHeight();  
 *// convert to 0-1* x /= w;  
 y /= h;  
 setZoom(**maxScale**, x, y);

}

Lại sử dụng transformCoordBitmapToTouch để chuyển focus pixel sang tọa độ trong layout với matrix mới . Sau đó đặt target\_hint\_im tại tọa độ đó

*// show target\_hint\_im and place it at touch point*PointF touch = **topImage**.transformCoordBitmapToTouch(pixelFocus.**x**, pixelFocus.**y**);  
placeViewInFrameLayout(**target\_hint\_im**, (**int**)touch.**x**, (**int**)touch.**y**);  
*// show scale animation while view is visible*scaleViewAnimation(**target\_hint\_im**, 0.5f, 2f);

Hàm thực hiện scale view animation

Sử dụng v.setVisibility(View.***GONE***) trong onAnimationEnd để đảm bảo view biến mất sau khi thực hiện xong animation

**public void** scaleViewAnimation(**final** View v, **float** startScale, **float** endScale) {  
 **onHintAnimation** = **true**;  
 **if** (v.getAnimation() != **null**) {  
 v.getAnimation().start();  
 **return**;  
 }  
 *// SET UP ANIMATION* Animation anim = **new** ScaleAnimation(  
 startScale, endScale, *// Start and end values for the X axis scaling* startScale, endScale, *// Start and end values for the Y axis scaling* Animation.***RELATIVE\_TO\_SELF***, 0.5f, *// Pivot point of X scaling* Animation.***RELATIVE\_TO\_SELF***, 0.5f); *// Pivot point of Y scaling  
 // anim.setFillAfter(false); // Needed to keep the result of the animation* anim.setDuration(1000);  
 anim.setRepeatCount(2);  
 anim.setRepeatMode(Animation.***REVERSE***);  
 anim.setAnimationListener(**new** Animation.AnimationListener() {  
 @Override  
 **public void** onAnimationStart(Animation animation) { }  
 @Override  
 **public void** onAnimationEnd(Animation animation) {  
 v.setVisibility(View.***GONE***);  
 **onHintAnimation** = **false**;  
 }  
 @Override  
 **public void** onAnimationRepeat(Animation animation) { }  
 });  
 v.startAnimation(anim);  
}

Tuy nhiên, ta gặp một vấn đề sau khi zoom tại điểm trung tâm block , animation scale đang hoạt động, nếu ta scroll image lúc này thì tọa độ của trung tâm thay đổi nhưng tọa độ của target\_hint\_im thì vẫn như cũ, nó khá thiếu tự nhiên như trong hình sau

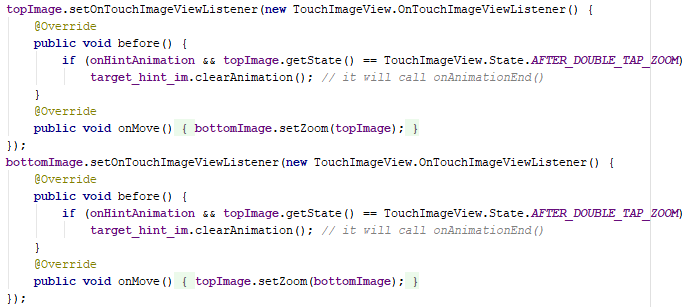


Vì thế ta cần target\_hint\_im biến mất ngay khi ta chạm vào TouchImageView

Ta sử dụng một biến **onHintAnimation** để kiểm tra animation có đang hoạt động không

Đồng thời kiểm tra tại thời điểm touch event bắt đầu bằng cách override phương thức before

trong OnTouchImageViewListener



**9. Demo**

Video chơi thử demo game , mỗi cấp độ 1 mission

Xem file record video demo.webm

