Candy Crush Solver

On ne démontre plus le succès du jeu mobile Candy Crush Saga : plus de 500 millions de téléchargements, et l'application nous rappelle à son démarrage que "Si on met bout à bout les déplacements réalisés chaque jour, on pourrait faire deux fois le tour du monde !". Chez Applidium, nous nous sommes demandé si l'on ne pouvait pas créer une application de résolution de Candy Crush, qui nous soufflerait les coups à jouer pour gagner. Un moyen de découvrir de nouvelles technologies intéressantes, tout en se penchant sur un jeu au succès mondial.

Mais nous ne sommes pas les premiers à creuser cette idée. Candy Crush est un problème NP-difficile, c'est-à-dire faisant parti des exercices mathématiques les plus ardus à résoudre. De nombreux joueurs se sont donc posé la question d'une solution automatisée. Un groupe d'élèves de l'IUT de Clermont-Ferrand a, par exemple, réalisé un travail de résolution à partir d'une version ordinateur du jeu. Grâce à une photo de leur écran, l'application Android affiche le meilleur coup à réaliser. D'autres ont créé une application mobile pour un jeu similaire (Puzzle and Dragons), mais il faut, avant de pouvoir l'utiliser, redessiner le niveau à la main. Nous souhaitions aller plus loin, en lisant directement la grille du téléphone, sans que l'utilisateur ait besoin de prendre une photo. Tout se ferait naturellement, dans le jeu. La reconnaissance de la grille sur Android, ainsi que l'affichage de la solution par-dessus le jeu, ont donc été des objectifs majeurs de notre recherche. Il aurait aussi été possible de résoudre Candy Crush sans passer par un seul niveau, comme le montre Stavros Korokithakis dans son étude des appels réseau entre l'application et le serveur. Mais cette technique nous éloignait de notre objectif : créer une aide pour le joueur, plus qu'un script de triche.

Nous avons alors essayé de développer ce "Candy Crush Cheater" sur Android. Le principe ? Une application qui analyse la grille de jeu, calcule le meilleur coup à faire et l'affiche directement sur l'écran, par-dessus Candy Crush.

Étapes

Notre application tourne en tâche de fond, pendant que le joueur déplace ses bonbons sur Candy Crush. Son fonctionnement original se reflète à travers ses cinq étapes principales.

L'accessibilité, une façon de suivre l'utilisateur

Notre application tourne en continu à partir du moment où le joueur démarre le service. Cependant, il serait vain de commencer à chercher les bonbons sur la home du téléphone, ou sur son fil d'actualité Facebook… Nous devons seulement agir lorsqu'il se trouve sur Candy Crush, et donc savoir exactement lorsque c'est le cas.

Nous utilisons les options d'accessibilité du téléphone pour savoir "où l'utilisateur se trouve". A chaque instant, nous pouvons recevoir le nom de l'activité en premier plan (mais aussi son contenu), à condition que l'utilisateur nous y autorise. Initialement créée dans le but de faciliter l'accès à toutes les applications pour les personnes souffrant d'un handicap visuel (lecture automatique du contenu), l'accessibilité nous permet en outre de suivre à la trace notre utilisateur. Big Brother is watching you ! Les applications autorisées à utiliser l'accessibilité de votre téléphone sont donc potentiellement dangereuses, car elles peuvent surveiller tous vos mouvements et le contenu de vos applications. Il est préférable de ne pas accorder cette permission trop facilement, pour ne pas mettre en danger vos données sensibles. Mais pour notre part, nous nous contentons de regarder si l'utilisateur se trouve ou non dans l'application Candy Crush.

Nous utilisons pour cela un `AccessibilityService`, capable de détecter tous les changement d'écrans du téléphone :

```
public class HeadService extends AccessibilityService {
   @Override
   public void onAccessibilityEvent(AccessibilityEvent event) {

if(event.getPackageName().toString().equals("com.king.candycrushsaga")){
        startImageListener();
     }
   }
}
```

Photographie de l'état actuel du jeu

Avant de pouvoir analyser l'écran du jeu, il faut déjà l'obtenir. C'est grâce à l'objet `MediaProjectionManager` que nous avons pu réaliser des captures d'écran régulières.

```
MediaProjectionManager mProjectionManager = (MediaProjectionManager)
   getActivity().getSystemService(Context.MEDIA_PROJECTION_SERVICE);
startActivityForResult(mProjectionManager.createScreenCaptureIntent(),
   REQUEST_CODE);
```

Le service réalise une capture dès qu'il repère un changement (grâce à `ImageReader.OnImageAvailableListener`). Pas besoin d'un téléphone rooté pour prendre un screenshot dynamiquement et l'enregistrer dans la mémoire de l'appareil ! Nous mettons ainsi à jour constamment l'image de référence pour le calcul du meilleur coup possible.

Analyse de l'image obtenue

Nous nous attaquons ensuite à la partie la plus coriace de notre application de résolution : la reconnaissance de la grille de bonbons.

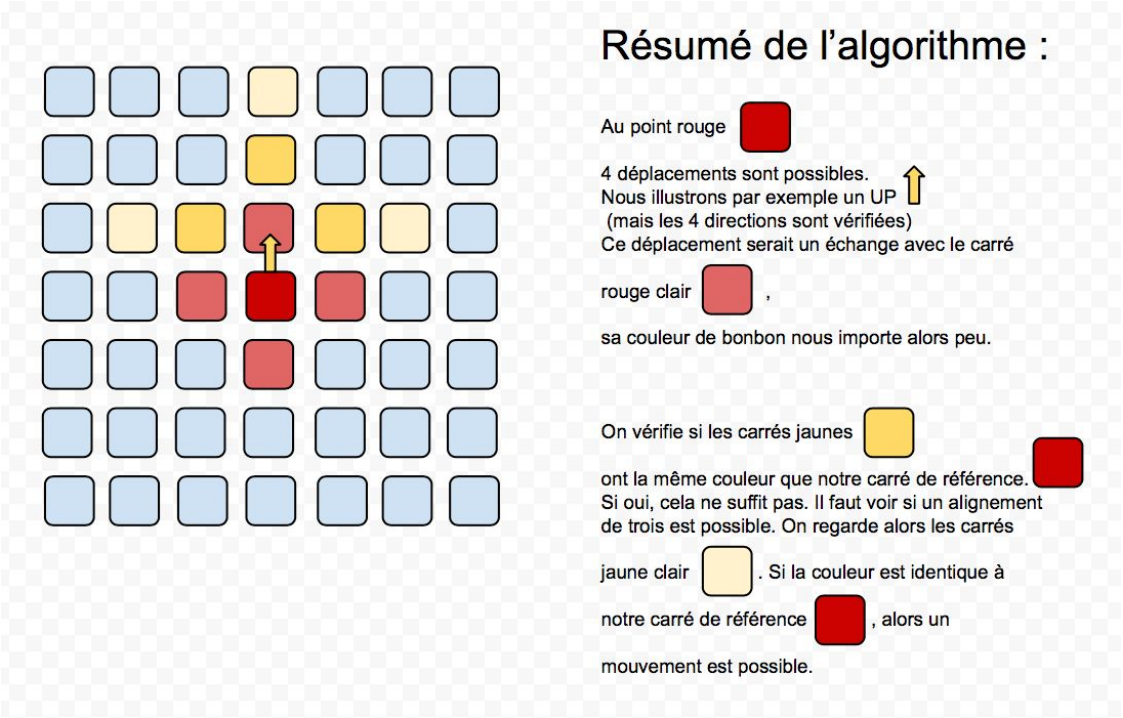
La méthode la plus efficace pour reconnaître un motif sur une image utilise sans aucun doute la convolution. Pour l'appliquer, nous prenons notre modèle (ici un bonbon coloré) et nous le déplaçons sur l'image, pixel par pixel, jusqu'à trouver l'endroit qui lui "correspond" le mieux (la correspondance se définit comme un score établi à partir d'une formule mathématique dépendant de la méthode de convolution choisie). Ce procédé est implémenté dans la bibliothèque OpenCV, sous le nom de `matchTemplate`. Après une premier essai avec cette méthode, qui donnait portant d'excellents résultats, nous nous sommes rendu compte que les performances des téléphones ne suffiraient pas à cette analyse et qu'il serait nécessaire de trouver une nouvelle solution.

Nous avons donc opté pour un algorithme plus simple ; notre service parcourt les pixels de l'image, jusqu'à trouver une couleur très proche du bonbon que nous recherchons. Une recherche simple par teinte, en oubliant les formes de nos modèles, nous a permis d'améliorer grandement les performances de l'application et de la rendre réellement utilisable (sans attendre quinze secondes avant de recevoir le déplacement à effectuer).

Recherche du meilleur coup à jouer

Maintenant que la grille est prête, nous pouvons la parcourir afin de trouver le meilleur coup à réaliser. Nous cherchons alors tous les mouvements possibles. La règle de Candy Crush est simple : un déplacement n'est possible que s'il permet l'alignement de trois bonbons de même couleur.

Notre algorithme de recherche parcourt la grille et regarde, pour chaque bonbon, 16 de ses voisins et examine leur couleur. Il détermine ainsi si un déplacement est possible vers le haut, vers le bas, vers la gauche ou vers la droite. Le schéma ci-dessous résume notre recherche :



En cherchant les déplacements, nous attribuons aussi des scores fictifs, afin de les classer les uns par rapport aux autres. Ils sont pour le moment assez intuitifs : 1 point pour un simple coup, 3 ou 4 points pour un alignement de 4 bonbons, 5 points pour la création d'un bonbon bombe, etc. Le but principal est de spécifier à l'algorithme une façon de classer ces déplacements, selon le nombre de points qu'ils nous offriront dans le jeu. Dans tous les cas, l'application se charge ensuite de trouver le meilleur et de l'afficher à l'utilisateur.

Affichage des résultats

Et nous arrivons à l'étape finale de notre application ! Étape qui nous permet de répondre à la question : "Pourquoi une application Android ? Une version bientôt disponible sur iOS ?" La réponse est non, car il faut afficher les résultats par-dessus Candy Crush. Montrer au joueur le déplacement à effectuer ne peut se faire sans cette fonctionnalité, disponible uniquement sur Android. On la retrouve par exemple avec les "bulles de conversation" Facebook, qui apparaissent à l'écran, par dessus n'importe quel autre contenu. C'est un peu ce que nous faisons sur l'écran de Candy Crush.

Une vue spécifique, équipée d'un WindowManager, est détenue par le service principal. Nous pouvons lui ajouter n'importe quel élément : un bouton, du texte, une image. Nous avons choisi cette dernière option et dessinons alors un contour assombri autour des bonbons à déplacer.

```
public class HeadLayer extends View {
          private     WindowManager.LayoutParams     params     =     new
WindowManager.LayoutParams(
              WindowManager.LayoutParams.WRAP_CONTENT,
              WindowManager.LayoutParams.WRAP_CONTENT,
```

```
                WindowManager.LayoutParams.TYPE_PHONE,
                WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE |
                WindowManager.LayoutParams.FLAG_WATCH_OUTSIDE_TOUCH,
                PixelFormat.TRANSLUCENT);

  private void addToWindowManager() {
                windowManager = (WindowManager)
                context.getSystemService(Context.WINDOW_SERVICE);
  windowManager.addView(frameLayout, params);

  LayoutInflater layoutInflater = (LayoutInflater)
                context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

  layoutInflater.inflate(R.layout.head, frameLayout);
  image = (ImageView) frameLayout.findViewById(R.id.solution);
}
```

Puis nous modifions ce contour en fonction des résultats suivants.

Quelques limites

Si ce travail nous offre une application Android fonctionnelle, elle reste néanmoins limitée sur certains points. Nous avons rencontré quelques obstacles tenaces, mais pourtant extrêmement intéressants pour la suite.

Des niveaux excessivement différents

L'application Candy Crush Saga rassemble plus de 2000 niveaux. Et une bonne dose d'objectifs différents ! Le gameplay sur lequel nous nous sommes concentrés concerne les premiers niveaux. Il faut marquer le plus grand nombre de points en un nombre de déplacements limités. L'application vise donc simplement le meilleur coup possible sur la grille. Mais pour d'autres niveaux, cette technique n'est absolument pas efficace. Les objectifs évoluent ; il faut alors faire tomber des fruits en bas de l'écran (et donc réaliser des déplacements sous ces fruits pour les faire descendre), ou éliminer des morceaux de gelée présents derrière les bonbons (et donc focaliser des déplacements sur ces endroits spécifiques). Et ces deux objectifs ne sont que des exemples. Une adaptation du Candy Crush Cheater est bien sûr possible. Nous avons pensé, par exemple, à demander à l'utilisateur l'objectif de son niveau avant de tenter de le résoudre. Mais cela implique de développer tous les moteurs correspondants. Un temps de production qui devient non négligeable.

Mais ce ne serait pas le seul obstacle. Avec les divers objectifs, les motifs varient aussi. Les 6 bonbons principaux reconnus par notre application ne suffisent alors plus. Il faudrait déceler tous les types de fruits possibles (cerises, marrons), mais aussi les morceaux de glace (étape à priori très difficile car ils se confondraient dans la transparence de l'arrière plan), les chocolats, réglisses, etc.

Un solver qui fonctionnerait pour toutes les nombreuses spécificités des niveaux de Candy Crush devient un projet beaucoup plus ambitieux.

Cependant, cet obstacle ne s'appliquerait pas à d'autres types de jeux mobiles : sudoku, Bejeweled Classic, Puzzle and Dragons, et tout autre jeu de logique. Nous pourrions donc imaginer nous tourner vers ce type de jeu, plus abordable car les objectifs n'évoluent pas avec le temps. La structure de notre application n'en serait alors pas modifiée : il suffirait de changer le moteur et de réorienter la reconnaissance des motifs. Une adaptation facilitée grâce aux étapes déjà implémentées.

Une mémoire vive en difficulté

Il a fallu faire face à un second problème : la mémoire vive disponible dans le téléphone. En effet, Candy Crush s'est révélé être une application extrêmement gourmande sur ce point,

occupant parfois une grande partie de la RAM. Si notre cheater a été optimisé au maximum, les calculs ne sont pas sans conséquence pour le hardware limité d'un téléphone.

Candy Crush est donc parfois ralenti par les recherches effectuées en arrière plan. Il est préférable de l'utiliser sur des téléphones puissants (même si des tests nous montrent qu'1Go de RAM suffit). Il est aussi souhaitable de fermer les autres applications actives (par exemple les "music player") avant de démarrer notre service, pour mettre l'appareil dans les meilleures conditions possibles.

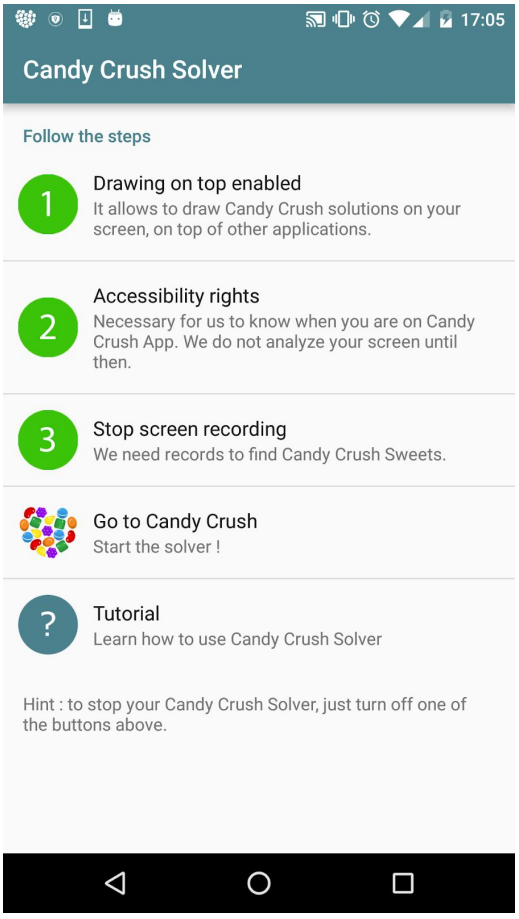
Si les performances du téléphone sont mises à rude épreuve, notre application reste parfaitement utilisable dans la plupart des cas. Nous pouvons aussi imaginer des améliorations en l'appliquant à d'autres jeux, moins gourmands en mémoire vive.

Conclusion

Candy Crush Cheater est donc une application qui se sert de nombreuses spécificités méconnues du téléphone et cette exploration a été une excellente façon de les découvrir. Si certaines fonctionnalités peuvent encore être améliorées, l'application est tout de même très efficace sur les principaux niveaux du jeu.

Cependant, nous souhaitons mettre en garde les utilisateurs contre ce genre d'applications. Candy Crush Cheater possède tout de même les droits pour capturer tous les mouvements de votre écran, lire le contenu textuel de votre mobile et écrire par-dessus vos applications. On peut alors imaginer de nombreux scénarios mettant en péril la vie privée ou la sécurité des joueurs. Un mot de passe ou un numéro de carte bleu présent en clair sur l'appareil devient tout à fait lisible et utilisable par l'application. De même, un contenu modifié peut être placé sur votre écran, sans que ce dernier soit détectable à vos yeux. Gardons alors en tête qu'il ne vaut mieux pas accepter toutes ces autorisations, à moins d'avoir une confiance sans faille en l'éditeur de l'application.

Enfin, nous fournissons le code source de l'application (lien), afin que chacun puisse se faire sa propre opinion du "Candy Crush Cheater", ou l'utiliser pour coder sa résolution automatique d'un autre jeu ! ;)


17:05
Candy Crush Solver
Follow the steps
1
Drawing on top enabled
It allows to draw Candy Crush solutions on your screen, on top of other applications.
2
Accessibility rights
Necessary for us to know when you are on Candy Crush App. We do not analyze your screen until then.
3
Stop screen recording
We need records to find Candy Crush Sweets.
Go to Candy Crush
Start the solver !
?
Tutorial
Learn how to use Candy Crush Solver
Hint : to stop your Candy Crush Solver, just turn off one of the buttons above.

Candy Crush Solver (English)

There is no need to tell you the huge success Candy Crush has had all over the world : more than 500 millions downloads and the application reminds us that “Players swipe more than two laps around the globe on their mobiles every day!”. Since the basic gameplay is pretty simple, we wondered at Applidium if we could make an application able to understand Candy Crush, and suggest the best moves, so that we can win easily. A way to discover new Android technologies, while working on a worldwide success game.

But we are not the first to think about it. Candy Crush is a NP-hard problem, which means that it belongs to the most difficult Maths exercices. Many players imagined an automated solution. For example, a little group of students from the Clermont-Ferrand University developped a solver for the computer game. It works by having a picture of a PC screen on which the game is played taken by an Android smartphone and the analysis is done on the smartphone. Other created a solver app for the same kind of game (Puzzle and Dragons), but first, you have to reproduce the level by hand. We would like to go further by reading the grid of sweets directly on the phone. Neither photo nor manual screenshot would be needed ! Everything would stay inside the same device. Mobile grid recognition and solution display on top have been our main goals while developing the app. It would also have been possible to solve Candy Crush without playing at all, as Stavros Korokithakis did, by studying network calls between the game and the server. But we are more interested in interacting with the game the same way a user would : via the screen.

So we tried to develop our Android “Candy Crush Cheater”. The idea ? An app that would recognize sweets in the game screen, compute the best move to make and overlay Candy Crush with the solution.

Steps

Our application runs in the background, while the player moves sweets on Candy Crush. Its behaviour can be divided in five main stages.

Accessibility, or how to follow your user

Our app is running continuously, from the moment the player starts the service. So we have to pay attention not to look for candies on his phone home screen or on his Facebook news feed. We should only search sweets when he is on Candy Crush, and therefore being aware when it happens.

We use accessibility options on the phone, in order to know precisely “where is our user?”. If the user gave us accessibility permission, we are able to learn the name of the foreground activity (and its content) at any moment. Whereas this feature was first created to help visually impaired people by reading aloud everything that is displayed on their screen, it also allows us to track our user. Big brother is watching you ! So accessibility applications are also a potential security issue. This insecurity comes from the fact that they can monitor people and access their screen content. We advise you not to give this permission too easily, to protect your personal and confidential data. But as far as our app is concerned, it only checks if the player is on Candy Crush.

The `AccessibilityService` is the perfect tool to do this, as it spots every screen change in your phone :

```
public class HeadService extends AccessibilityService {
   @Override
   public void onAccessibilityEvent(AccessibilityEvent event) {

if(event.getPackageName().toString().equals("com.king.candycrushsaga")){
        startImageListener();
     }
   }
}
```

Freeze the current game state, with a picture

We cannot analyze the candy grid in the screen if we don’t have a record of the screen yet. The object `MediaProjectionManager` allows us to take screenshots regularly.

```
MediaProjectionManager mProjectionManager = (MediaProjectionManager)
   getActivity().getSystemService(Context.MEDIA_PROJECTION_SERVICE);
startActivityForResult(mProjectionManager.createScreenCaptureIntent(),
   REQUEST_CODE);
```

The service change the screenshot each time he finds a change in the screen (thanks to a `ImageReader.OnImageAvailableListener`). No need for a rooted phone to make screen records programmatically and save them in the internal storage ! We update the reference image continuously, so that the computed solution can be reliable.

Extracting sweets from the screenshot

Here comes the hardest part of our Candy Crush work : the image recognition.

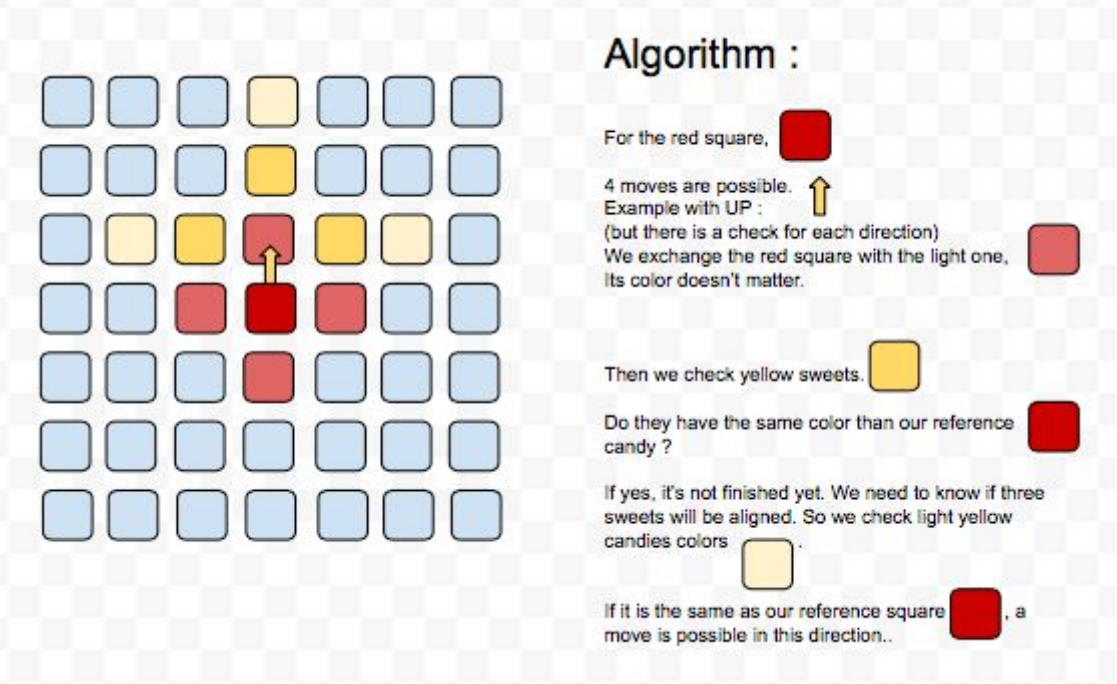
Convolution is, without a doubt, the best way to find a template in a bigger picture. The principle is simple : we take our little feature and we move it, pixel by pixel, alongside the sample image. Each pixel is given a score, according to a math formula, determined in advance (depending on the chosen method of convolution). The bigger the score is, the best the current set of pixels matches with the expected feature. In the end, we simply keep the pixel with the best score. That is how we can extract templates with convolution, and in our case, extract sweets from the screenshot. This algorithm exists in the library OpenCV, and is called `matchTemplate`. We tried using this for a long time (and it was giving us excellent results). However, the mobile performances are very limited and we realized that this kind of convolution would be too greedy. Our application was really slow and we could not continue this way.

So we chose a simpler algorithm. Our service scans the image and concentrates on pixel colors. If it is really close to the green candy color, we save it as a green candy. And so on for every sweet color. Searching by hue instead of searching by shape helped us to improve the performance and the application speed. With this important progress, our Candy Crush Cheater became really usable. No need to wait fifteen seconds before obtaining a solution.

In search of the best move

The grid is now ready and we can browse it to find what we should play. First, we look for every valid move, meaning that it should create a line of three candies.

Our algorithm takes each sweet in the grid and looks at its neighbours' colors. Thereby it is able to know if an up move (or whatever direction) is possible. This picture shows how it works :



Rather than just listing possible moves, we assign them a fictive score, so that it is possible to know which one is better than the other. They are very intuitive for now : 1 point for a plain move, 3 or 4 points for a line of 4 candies, 5 for a bomb creation, and so on. We only want to rank moves, according to the number of points they will really offer the player in the game. In any case, the application finds the best move and displays it for the player.

Overlaying the top result

Last step before getting the solution! But we still didn't answer the question : "Why an Android app? Will it be released in iOS soon?". Well, no. Because we need a special feature, only available on Android phones : displaying content on top of every other apps. Facebook uses this unit with its Messenger chat heads, for example. In our solver, we apply it as a way to show our solution on top of Candy Crush and show it to the player.

The main service owns a specific view, equiped with a WindowManager. Any item can be added to this special view, as a text, a button or a picture. We chose an ImageView to draw the borders of the two candies the player needs to swap.

```
public class HeadLayer extends View {
           private     WindowManager.LayoutParams     params     =     new
WindowManager.LayoutParams(
                WindowManager.LayoutParams.WRAP_CONTENT,
                WindowManager.LayoutParams.WRAP_CONTENT,
                WindowManager.LayoutParams.TYPE_PHONE,
                WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE |
                WindowManager.LayoutParams.FLAG_WATCH_OUTSIDE_TOUCH,
                PixelFormat.TRANSLUCENT);

  private void addToWindowManager() {
                windowManager = (WindowManager)
                context.getSystemService(Context.WINDOW_SERVICE);
  windowManager.addView(frameLayout, params);

  LayoutInflater layoutInflater = (LayoutInflater)
                context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

  layoutInflater.inflate(R.layout.head, frameLayout);
  image = (ImageView) frameLayout.findViewById(R.id.solution);
}
```

Then we update the ImageView content, according to the latest results.

Difficulties encountered

All this work leaded to a functional app ; however it remains limited on some thorny issues. We faced tenacious problems, but they appeared to be extremely interesting for further evolutions.

Too many levels

"There are almost 2000 levels in Candy Crush Saga", reminds the little pink girl when we open the app. 2000 levels, and almost as many different objectives! We focused on the main goal of first levels : making the best score in a limited number of moves. Our Candy Crush Cheater aims simply to find the best scoring move on the current grid. But this algorithm is absolutely useless for some levels. And objectives evolve over time, as new levels are still being created. For example, in a level, we have to drop fruits on the bottom of the grid (and so the algorithm should tell us to make moves just below them). In another level, we have to remove jelly parts behind sweets (and focus our moves on these candies to break them). And we can find a lot more of them. A Candy Crush Cheater improvement is still possible : we could ask the player which objective will appear on the level she picked. However, it means that we have to develop every possible engine, for every single goal. The developing effort becomes significant.

But this is not the only problem with growing number of levels. With them, the number of patterns is also growing. We recognize the six main candies, but it won't suffice any more. We would have to look for fruits (cherries, chestnuts), jellies (not an easy step as they can be mixed with the background), chocolates, licorice…

A solver that would take all these features into account would become a much more ambitious project. Nevertheless, we would not face these obstacles on other mobile games : Sudoku, Bejeweled Classic, Puzzle & Dragons, or every kind of logic game that has a more focused set of possible objects. We could imagine adapting our solver - and it would not be difficult at all ! Our architecture would not change, like most of the code. The changes would only modify the engine (creating a new algorithm to solve the game) and the pattern recognition (features extraction). An easy adaptation !

RAM in trouble

We unfortunatly faced a second problem : available RAM on phone. Indeed, Candy Crush is already very greedy, to which we add our background service, which is computing to find

patterns. Even if we optimized our calculation, the limited mobile hardware is sometimes not ready for this.

Then Candy Crush is occasionally slowed by our background running service. It's still better to use on recent phones (even if tests on 1Go RAM phone perfectly succeeded). We also advise you to close other active background apps, such as music players for example, so that the device is under proper conditions to work with Candy Crush and its solver.

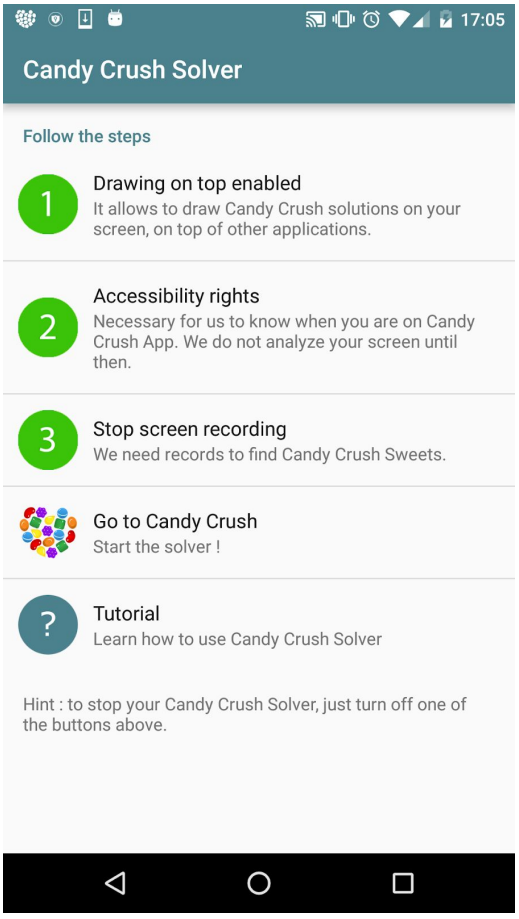
Even if we strained the phone performances, our app is totally usable most of the time. We could still imagine better results on less greedy games.

Conclusion

Candy Crush Cheater is an app that uses a lot of new Android specificities and it was very interesting to discover them while developing our solver. Even if some features could still be improved, the app remains very efficient on the first levels of the game.

However, we would like to insist on the fact that this kind of apps remain extremely dangerous. Candy Crush Cheater has permissions to access every textual content on your phone, take continuous screenshots, and write on top of every other apps. Privacy and safety of players could be in danger. Plaintext password or credit card number could be perfectly readable by the application. Likewise, modified text or content can overlay your screen, in a way that you do not notice it. So let's keep in mind that we should not give these permissions too easily. Trust the editors of the app is a necessary condition before accepting.

Finally, our source code can be found on Github (link), so that everyone can use it to code his own automated solver and help players on other games ! Don't hesitate to check it anyway, to see it all by yourself ;)



Code that helped us

https://github.com/mollyIV/ChatHeads

https://github.com/mtsahakis/MediaProjectionDemo

https://github.com/JakeWharton/ViewPagerIndicator