Tagarela

Jogo de letras/números voltado para tecnologia assistiva no Android

Wagner Jean Reetz
Orientador: Dalton Solano dos Reis

FURB – Universidade Regional de Blumenau
Grupo de Pesquisa em Computação Gráfica, Processamento de Imagens e Entretenimento Digital
www.inf.furb.br/gcg/tagarela

Tagarela

Roteiro

- Introdução
- Objetivos
- Fundamentação teórica
- Desenvolvimento
- Resultados e discussões
- Conclusão
- Extensões

Tagarela

Introdução

- Dispositivos móveis
  - Evolução
  - Mercado
  - Plataforma Android
  - Jogos
- Tecnologia assistiva
  - Autonomia
  - Independência
  - Qualidade de vida
  - Inclusão social
- Projeto Tagarela

Tagarela

Objetivos

- Disponibilizar um jogo 2D que manipula letras e números
- Especificar uma arquitetura com modelagem orientada a componentes
- Propiciar um ambiente que auxilie o desenvolvimento e evolução de pessoas com necessidades especiais e crianças em fase de alfabetização
- Disponibilizar cenários utilizando uma variedade de imagens e áudios

Fundamentação teórica

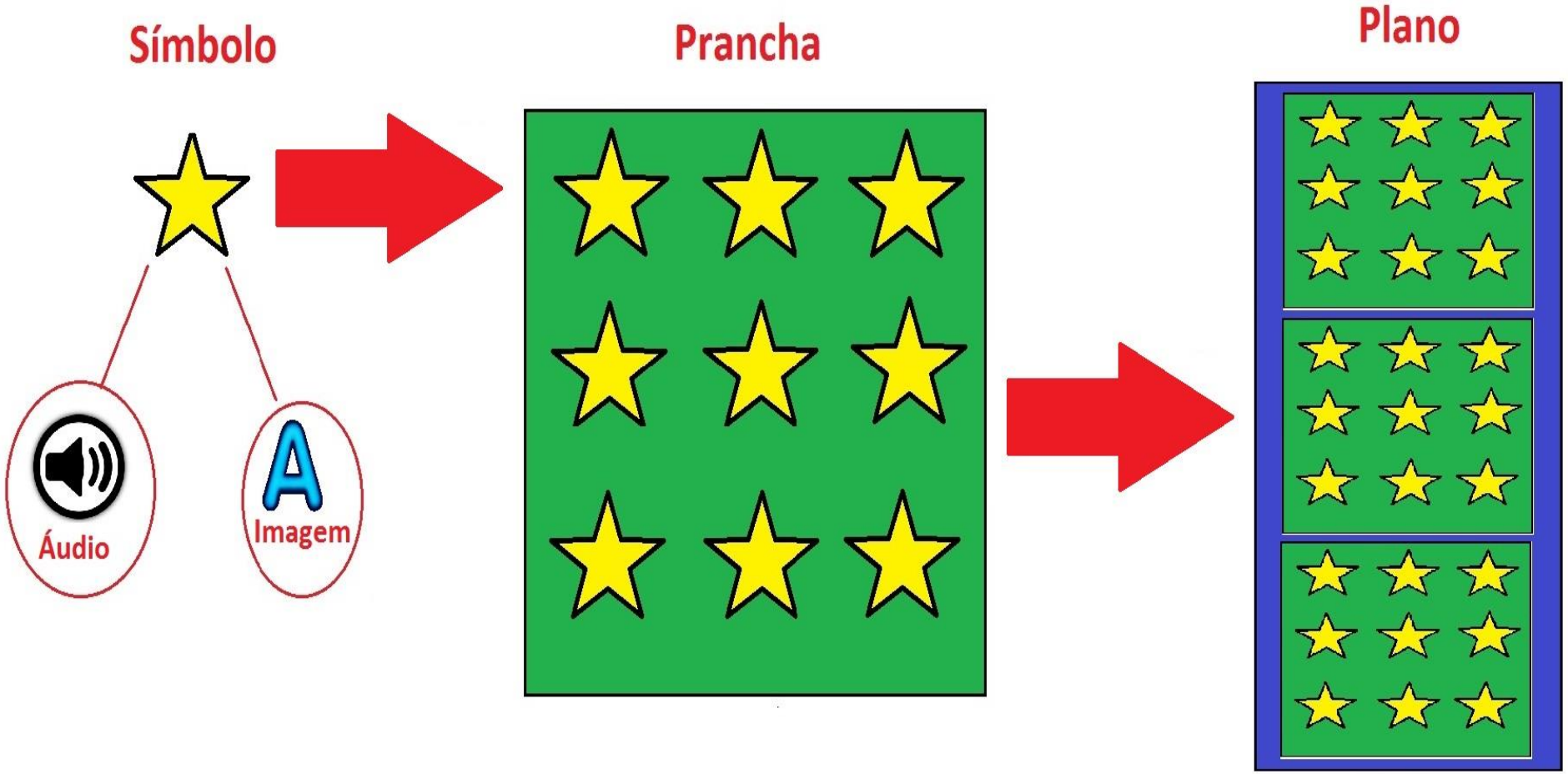
Planos, pranchas e símbolos

- Planos: agrupamentos de pranchas que possuem mesma finalidade de aprendizado
- Pranchas: agrupamentos de símbolos que possibilitam o tutor interagir com o paciente para propiciar uma evolução na capacidade de comunicação
- Símbolos: compostos por uma imagem que representa uma ação, local, evento ou figura qualquer, e também um áudio para assimilar a imagem

Fundamentação teórica

Planos, pranchas e símbolos

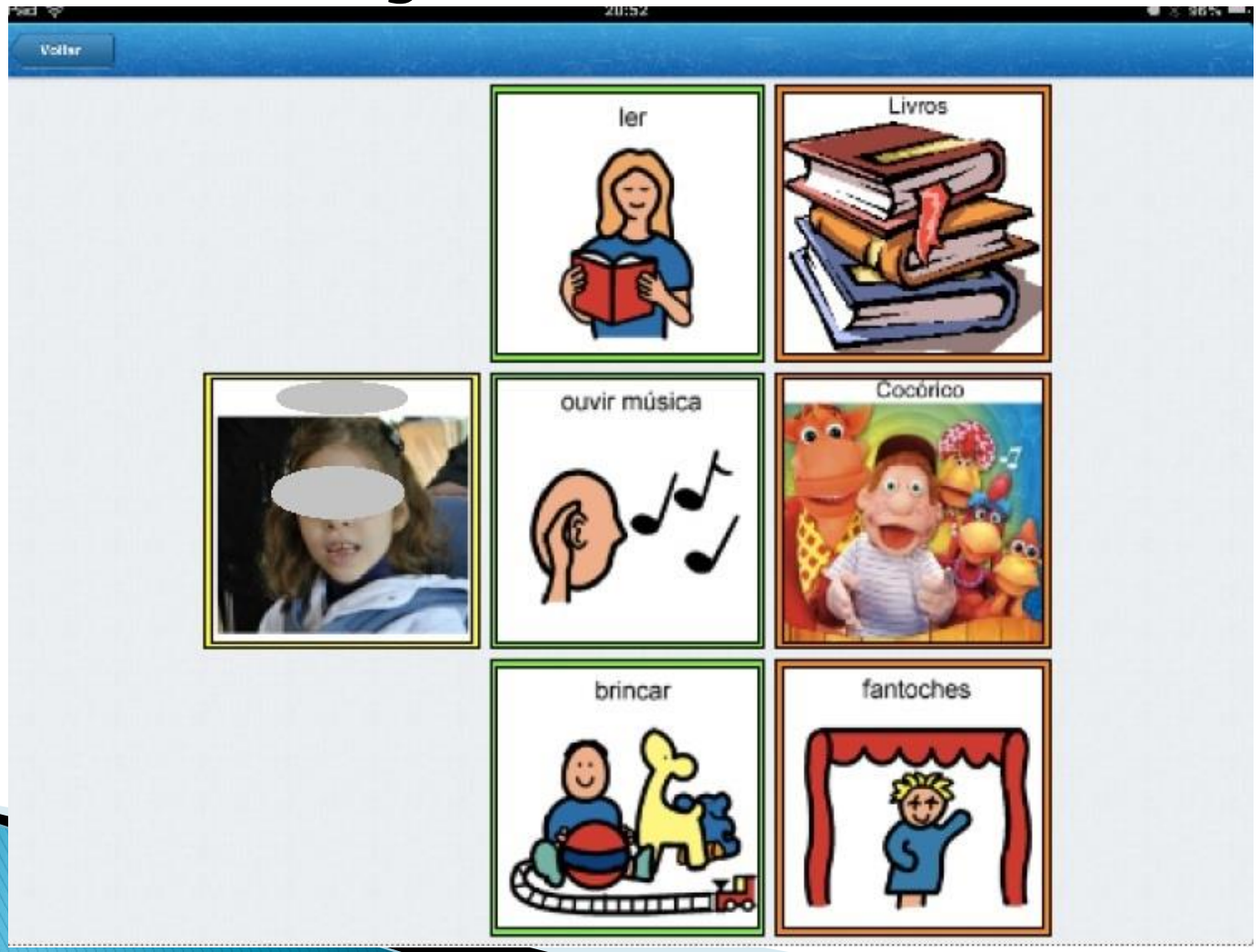
- Exemplo de estrutura:



Fundamentação teórica

Planos, pranchas e símbolos

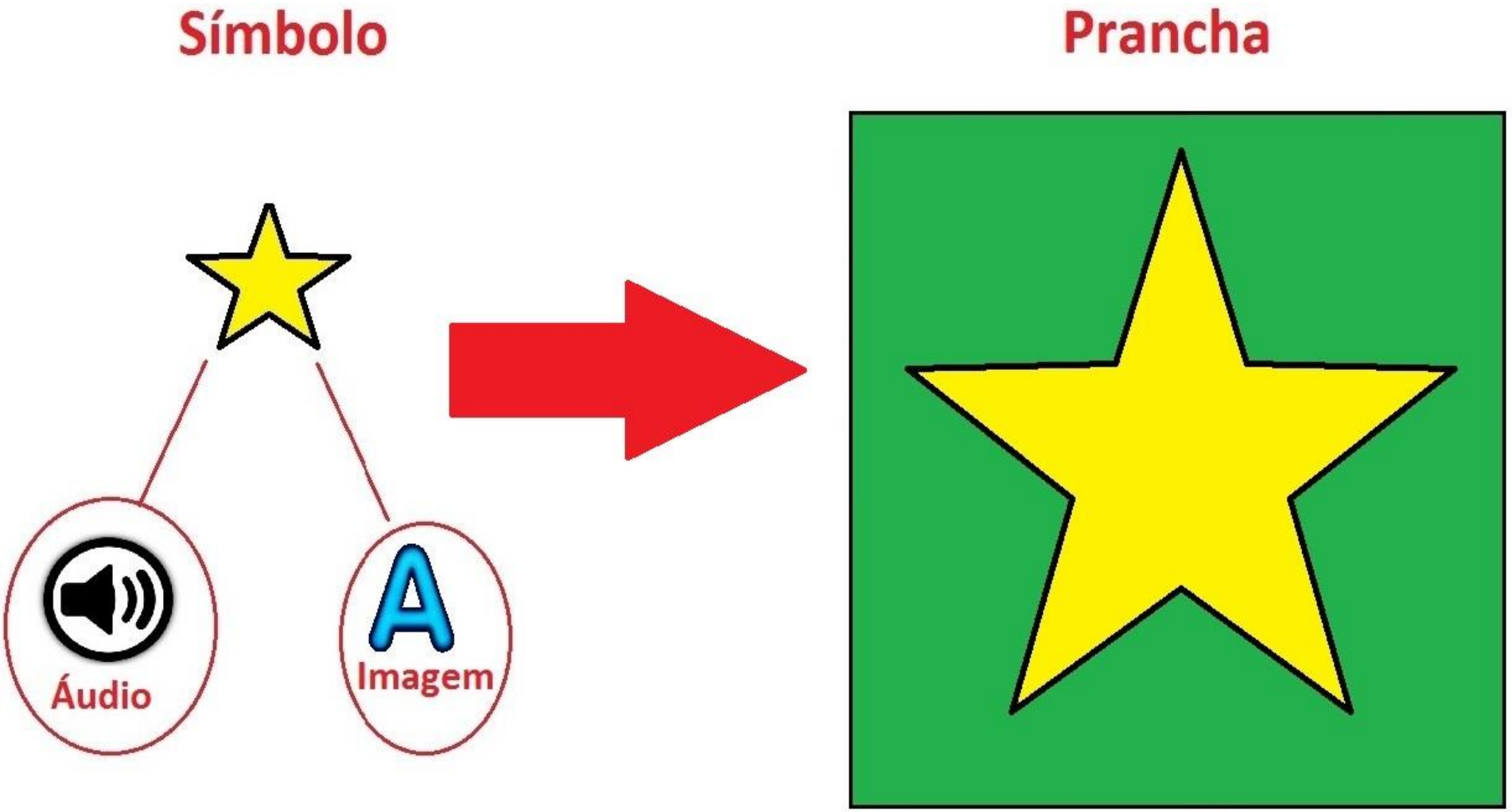
- Prancha no Tagarela:



Fundamentação teórica

Planos, pranchas e símbolos

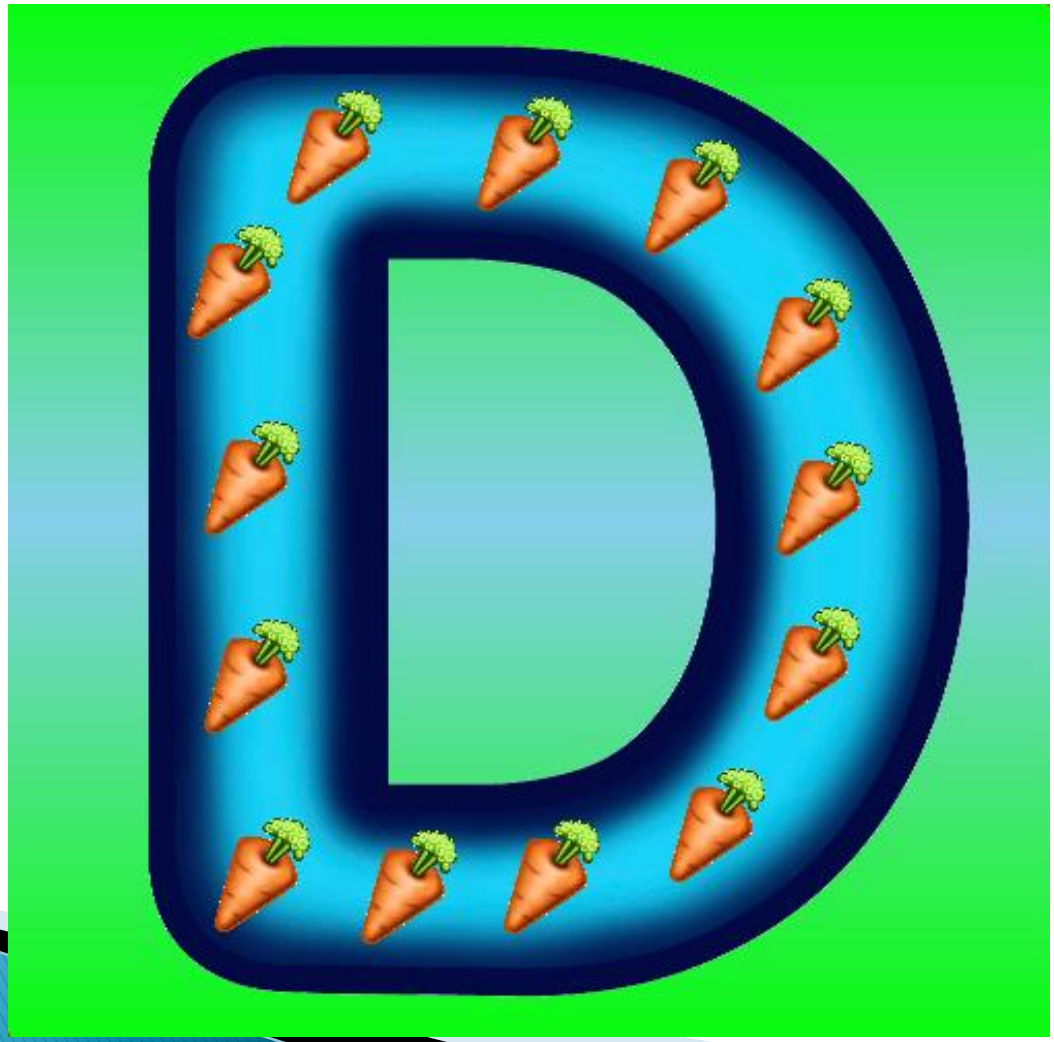
- Prancha neste trabalho:



Fundamentação teórica

Planos, pranchas e símbolos

- Prancha neste trabalho:



Fundamentação teórica

Caçador e presa

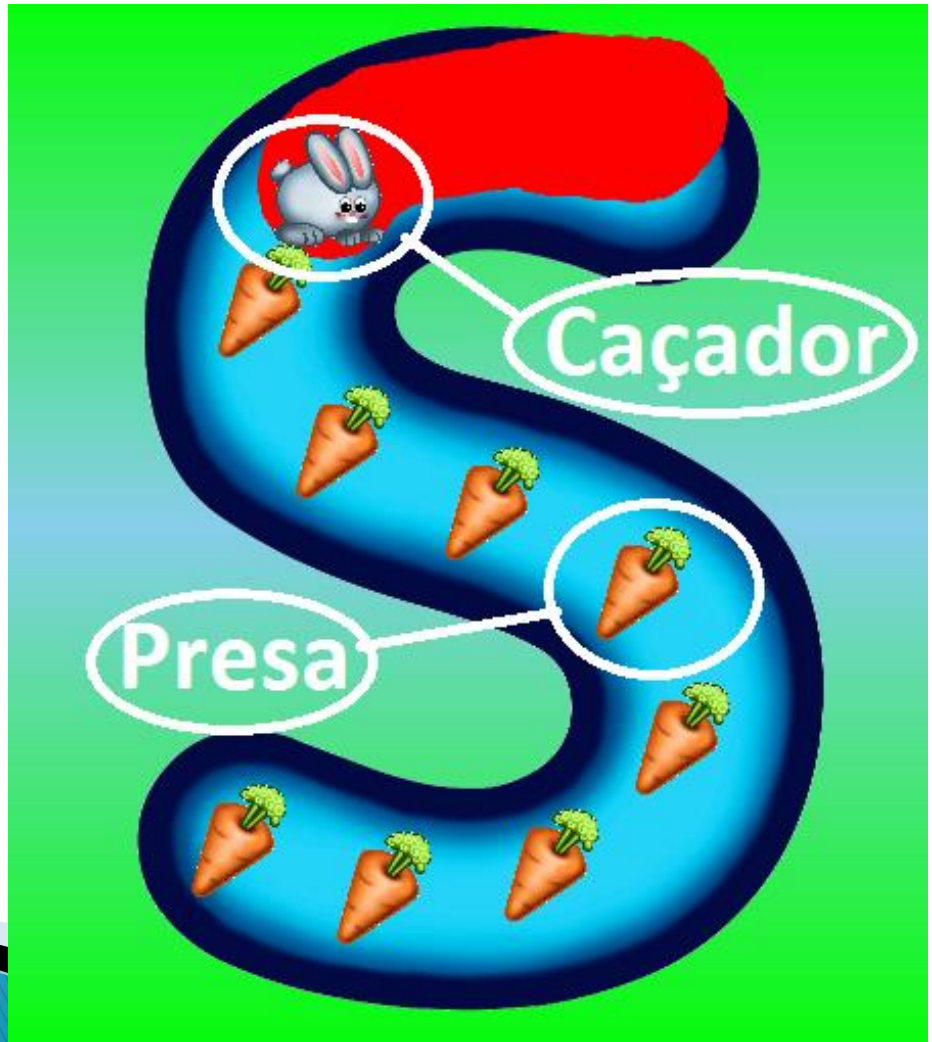
- Caçador: representa a imagem que segue o toque do usuário na tela
  - Exemplos:

- Presa: representa a imagem de um ponto de interesse
  - Exemplos:

Fundamentação teórica

Caçador e presa

- Exemplo:



Fundamentação teórica

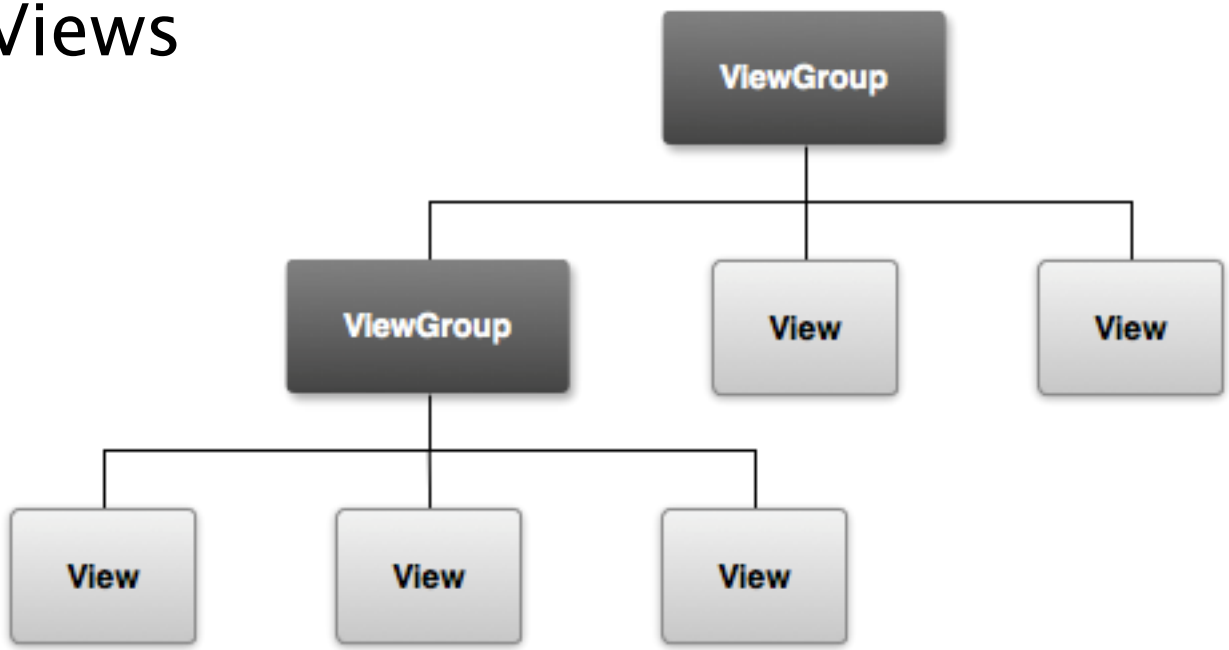
Modelagem orientada a componentes

- Unidade de software independente
- Implementação e projeto encapsulado
- Interfaces para comunicação externa
- Parte substituível
- Ter uma função clara e objetiva

Fundamentação teórica

Plataforma Android

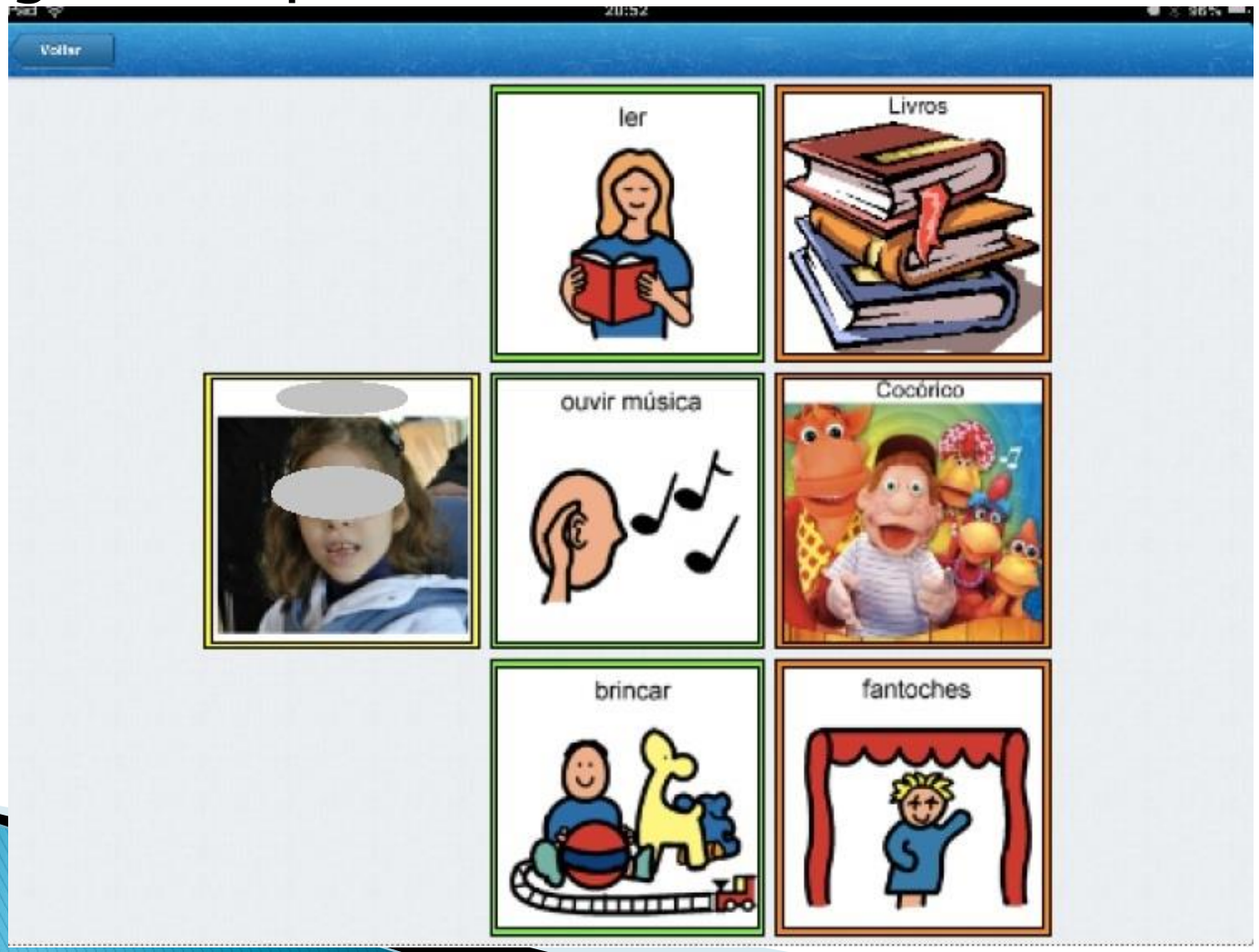
- Activities: representa uma tela na aplicação
- View: classe base para componentes de *User Interface* (UI)
- ViewGroup: representa um *layout*, isto é, um *container* de Views



Fundamentação teórica

Trabalhos correlatos

- Tagarela: aplicativo PCA no iOS



Fundamentação teórica

Trabalhos correlatos

- Dibugrama:

Fundamentação teórica

Trabalhos correlatos

- Desenhe e Aprenda a Escrever:

Desenvolvimento

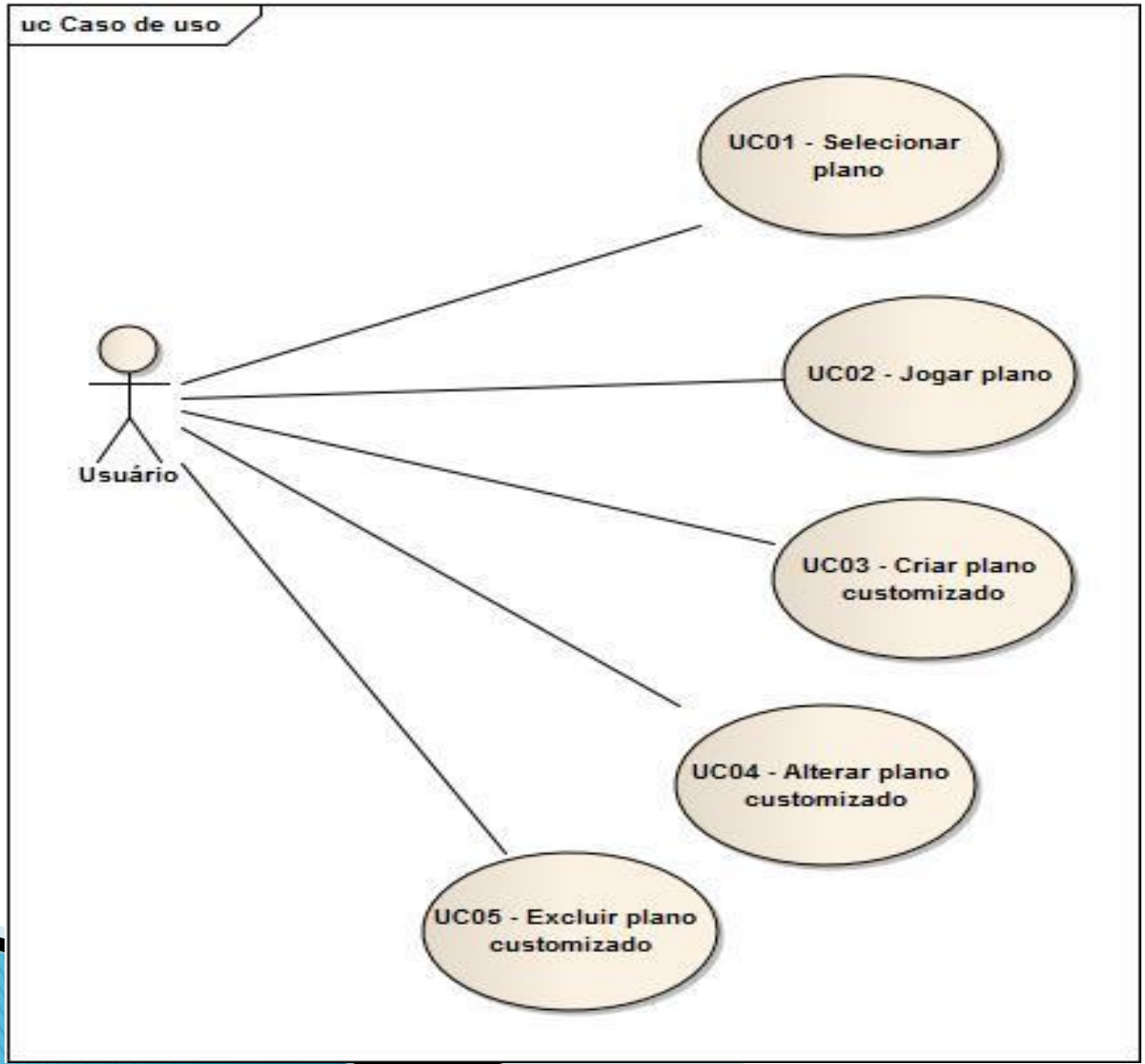
Requisitos funcionais

- utilizar cenários e atividades, que permitam trabalhar com todas as letras e números da língua portuguesa
- permitir que o usuário interaja com as atividades através do toque
- permitir que o usuário crie planos customizados
- permitir que o usuário altere planos customizados
- permitir que o usuário exclua planos customizados
- apresentar ao usuário o áudio do símbolo quando a prancha for concluída

Tagarela

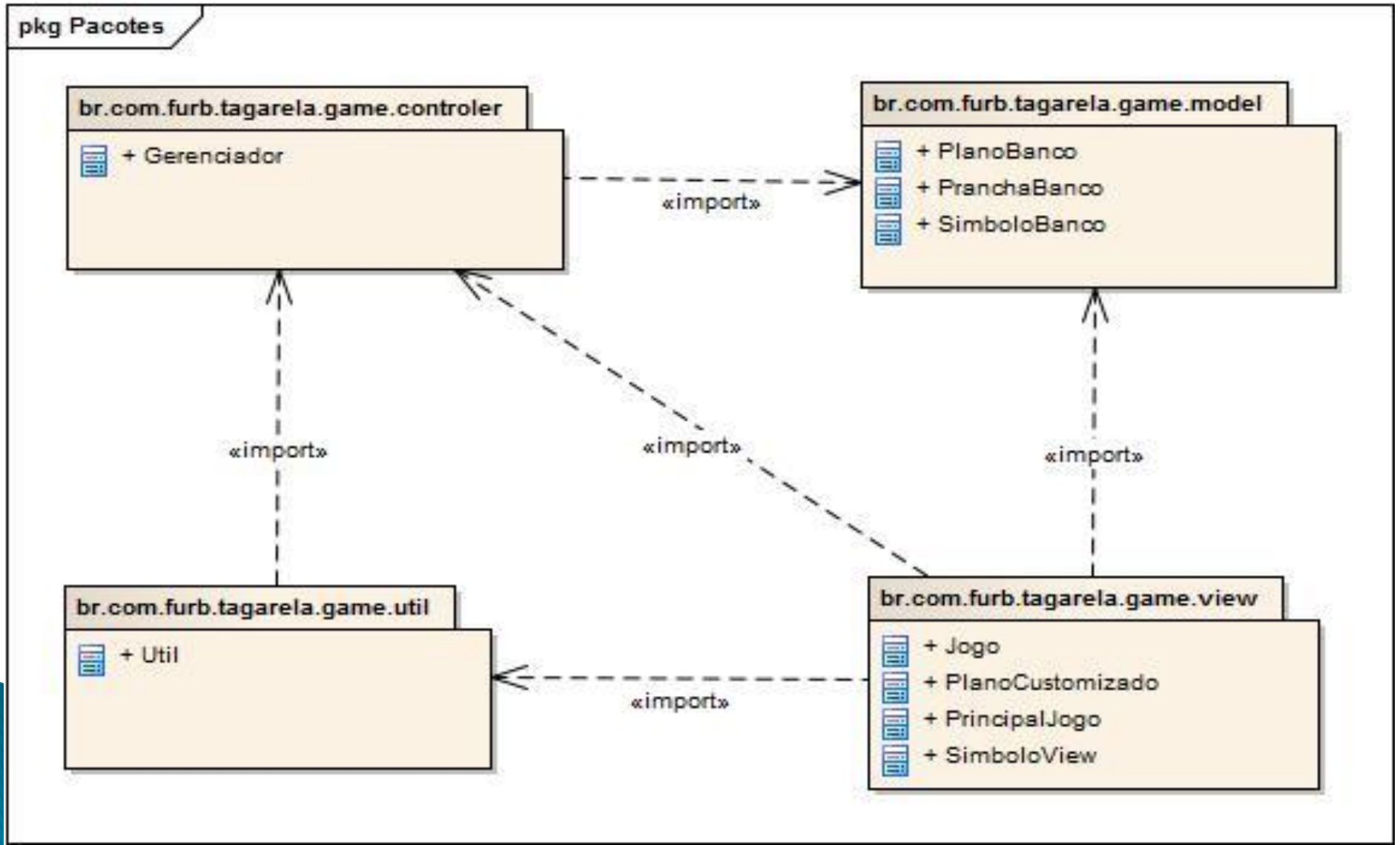
Desenvolvimento

Casos de uso



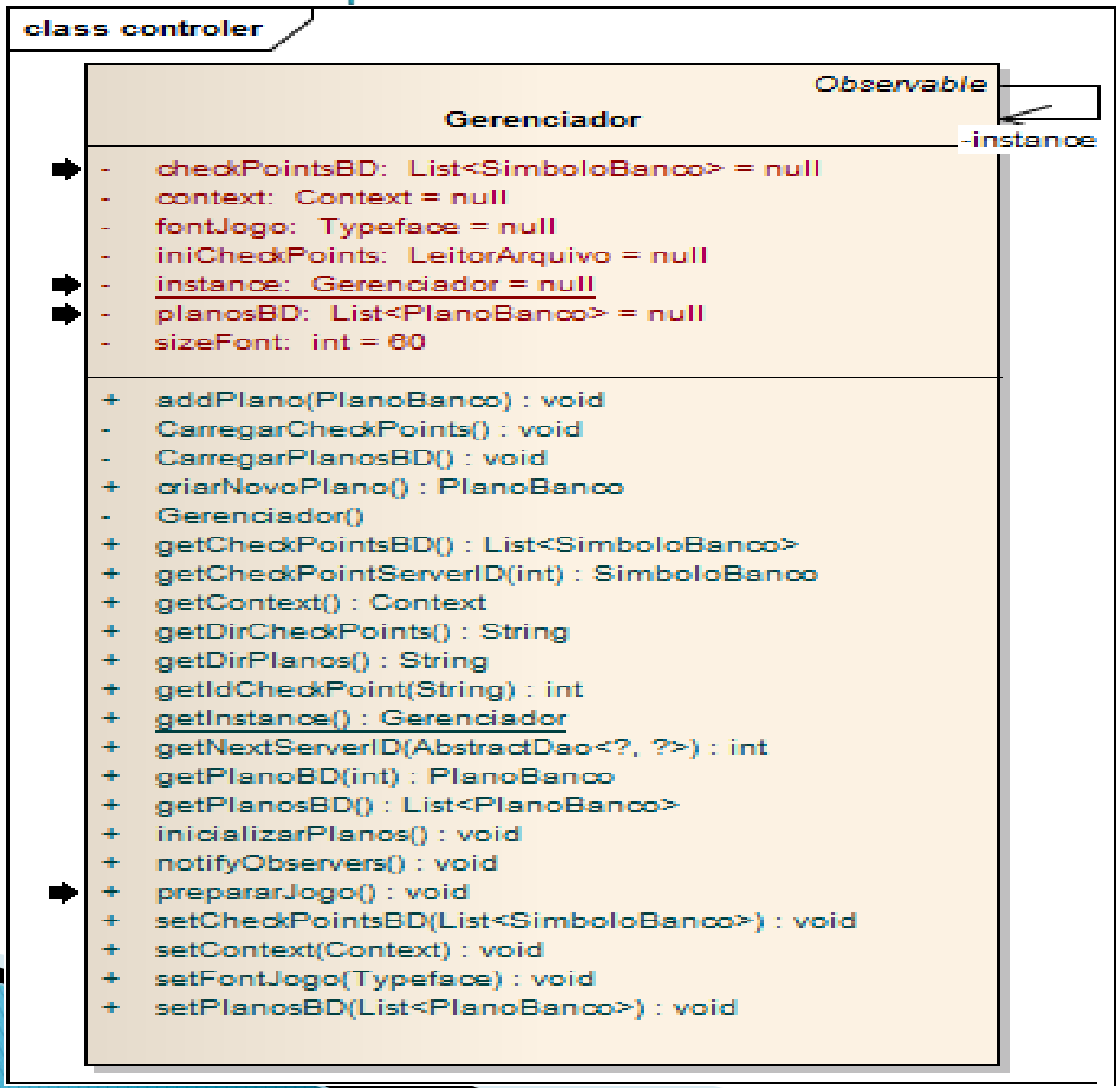
Desenvolvimento

Diagrama de classes: Pacotes



Desenvolvimento

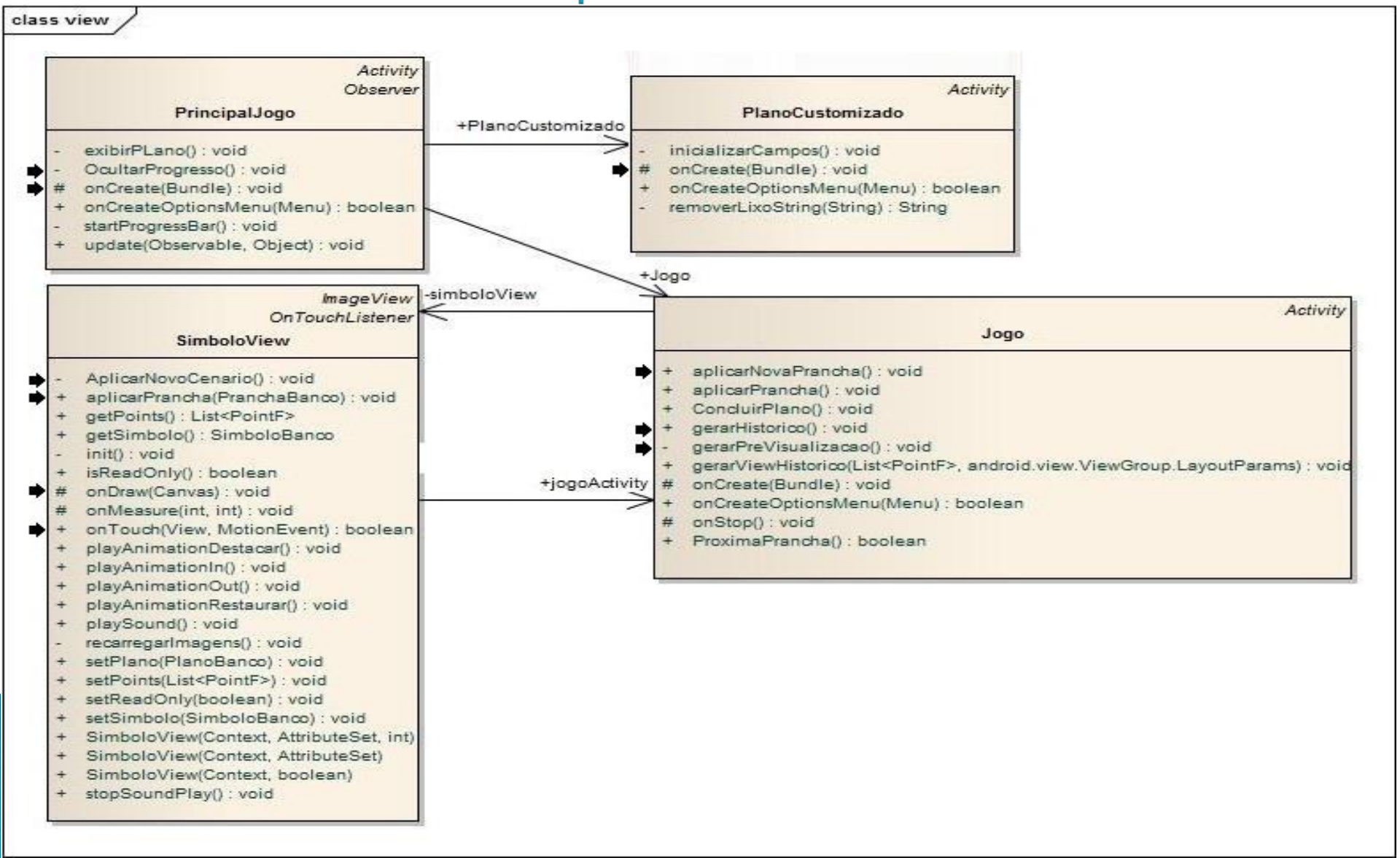
Principais classes: controler



Tagarela

Desenvolvimento

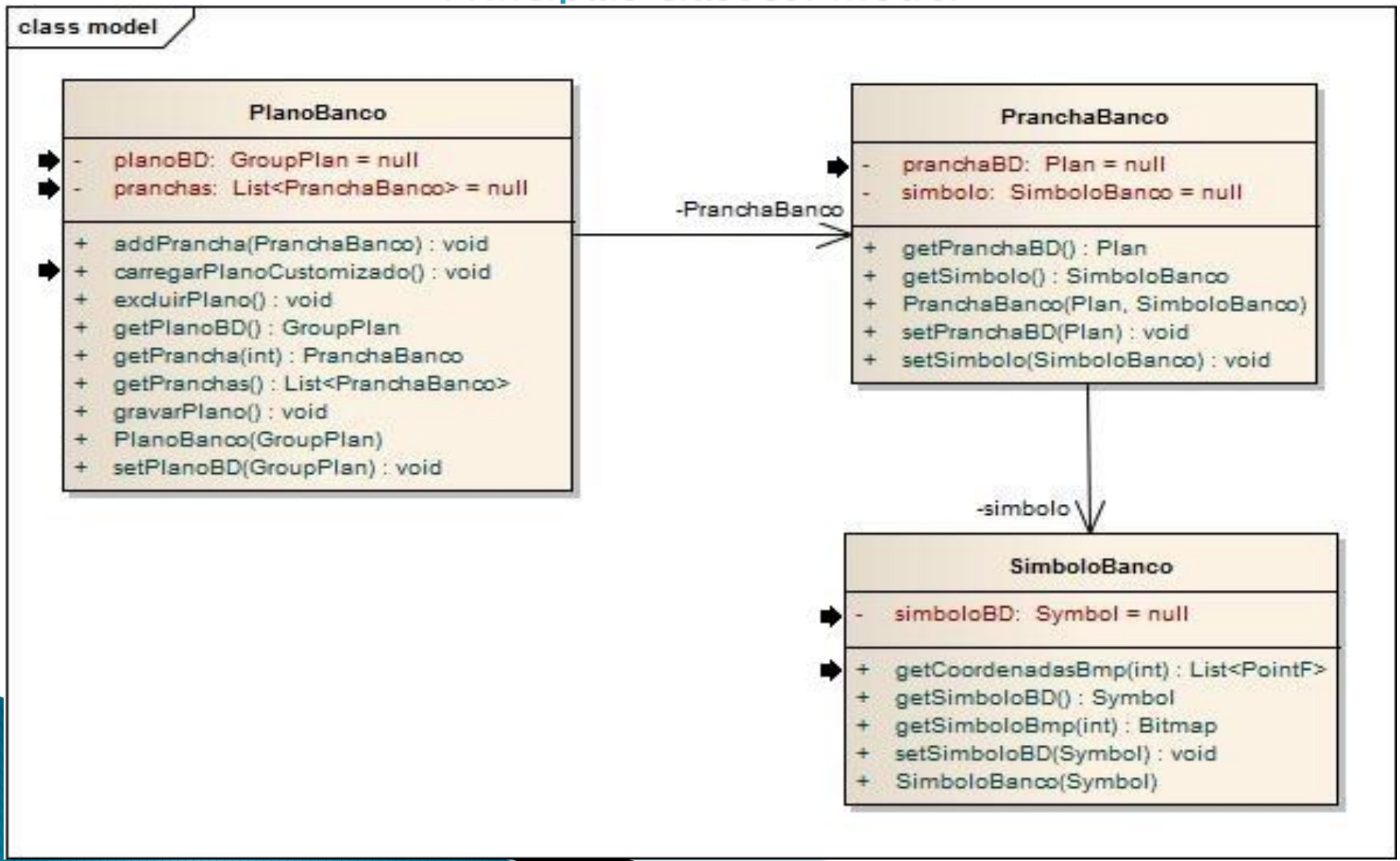
Principais classes: view



Tagarela

Desenvolvimento

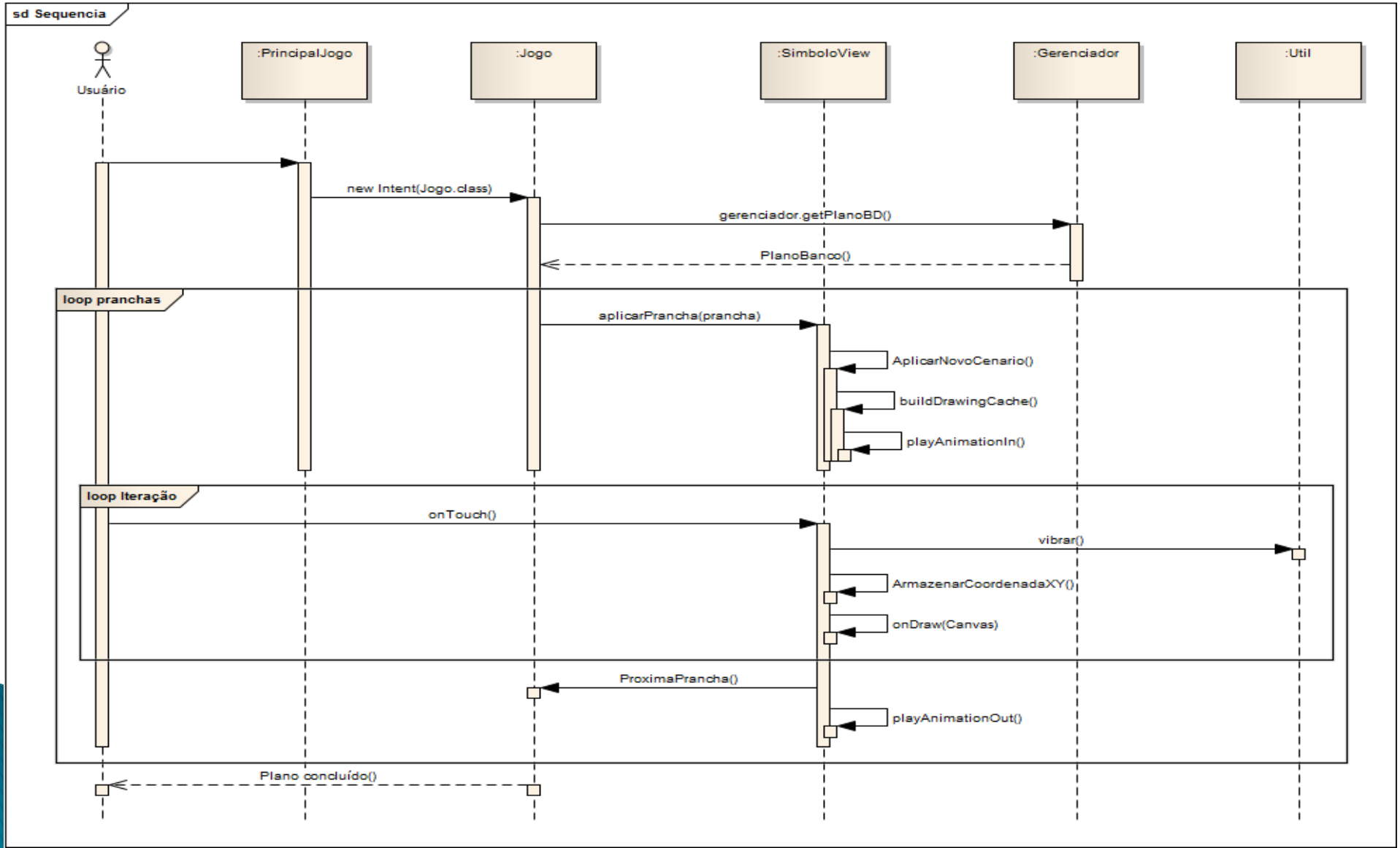
Principais classes: model

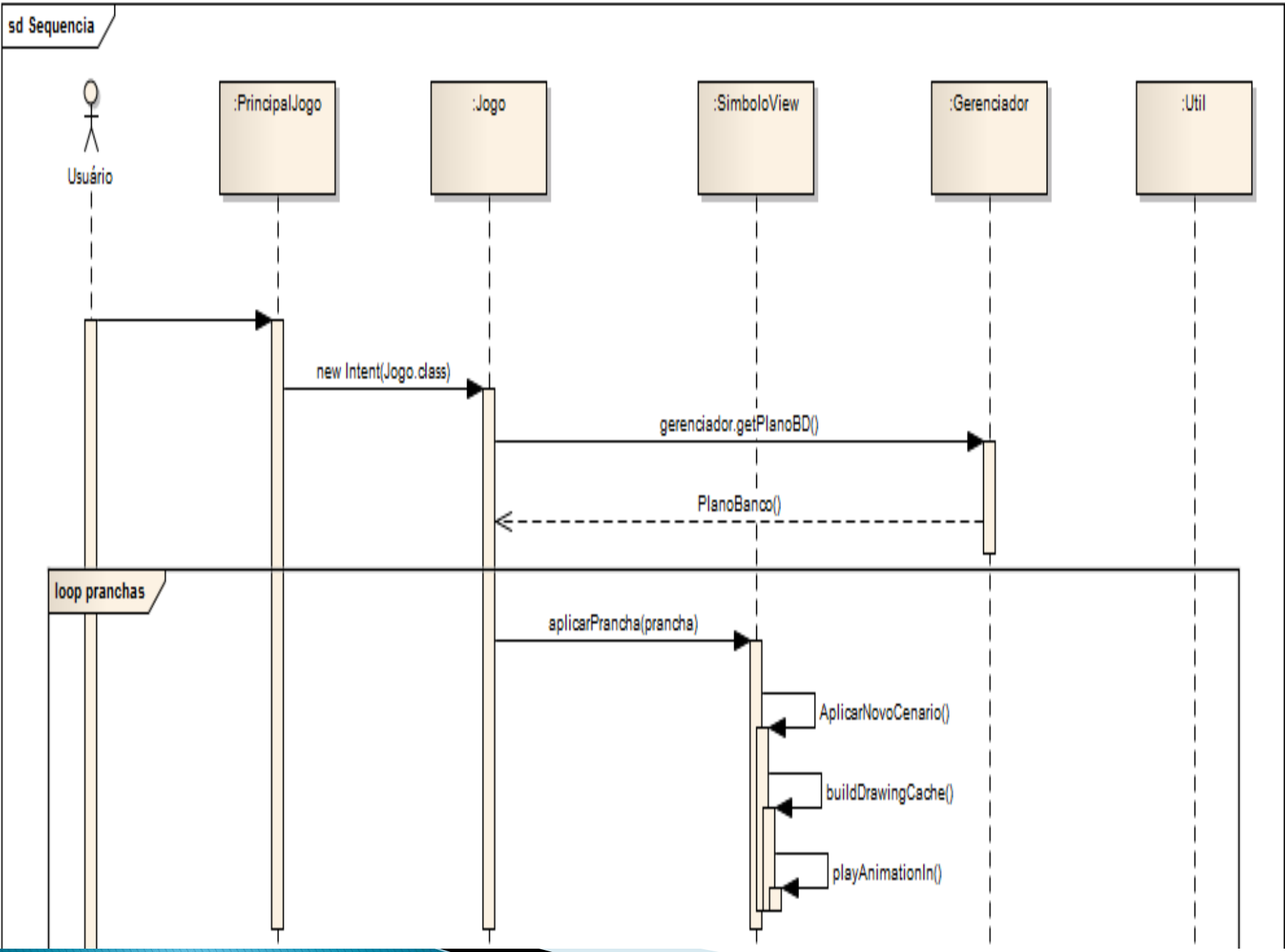


Tagarela

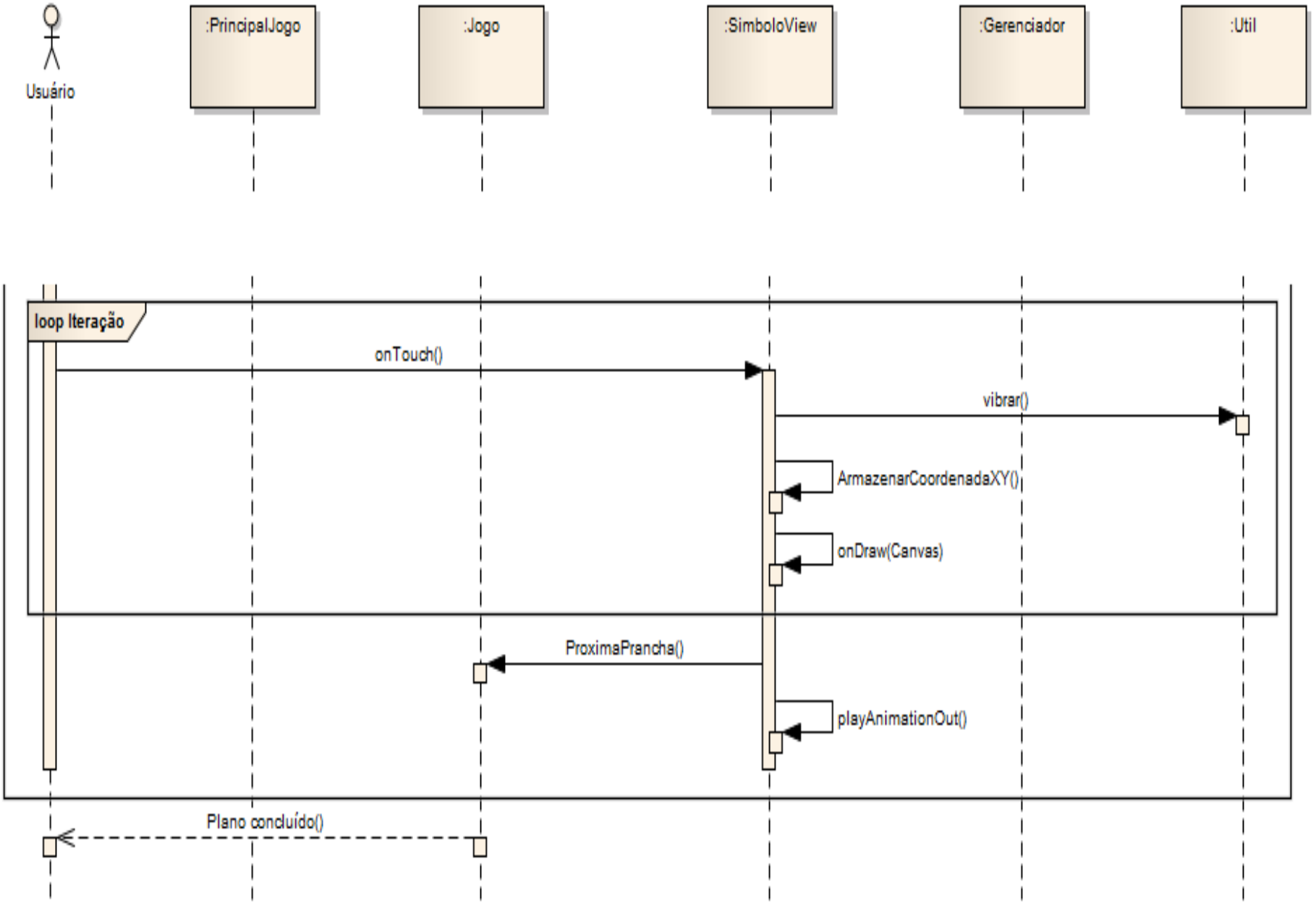
Desenvolvimento

Diagrama de sequência




sd Sequencia
Usuário
:PrincipalJogo
:Jogo
:SimboloView
:Gerenciador
:Util
new Intent(Jogo.class)
gerenciador.getPlanoBD()
PlanoBanco()
loop pranchas
aplicarPrancha(prancha)
AplicarNovoCenario()
buildDrawingCache()
playAnimationIn()

Continuação...



Tagarela

Desenvolvimento

Técnicas e ferramentas utilizadas

- Técnicas utilizadas:
  - Orientação a objetos
  - Padrões de projetos
    - *Singleton*
    - *Observer*
- Ferramentas utilizadas:
  - Eclipse 4.2.1
  - Enterprise 7.5
  - Android *Development Tools* (ADT)
    - *Plugin* do Eclipse
  - Adobe Photoshop CS5
- Tablet Samsung Galaxy Note 10.1

Desenvolvimento

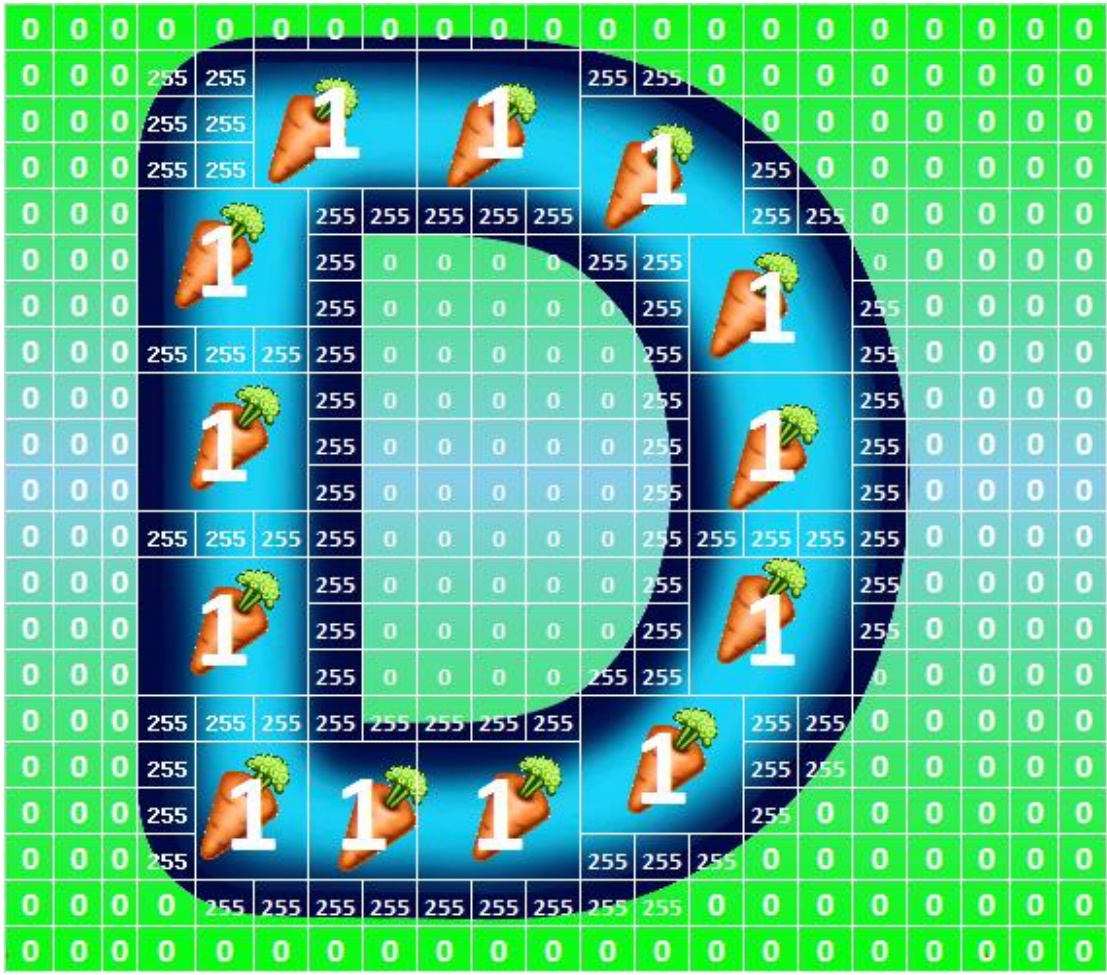
Implementação: Localizar pontos de interesse

```
34  public List<PointF> getCoordenadasBmp(int tamanho){
35        Bitmap bmp = getSimboloBmp(1000);
36
37        List<PointF> points = new ArrayList<PointF>();
38
39        for (int i = 0; i < 1000; i++) {
40              for (int j = 0; j < 1000; j++) {
41                    int color = bmp.getPixel(i, j);
42                    int alpha = Color.alpha(color);
43
44                    if ((alpha > 0) && (alpha < 255)) {
45                          PointF p = new PointF(i, j);
46                          p.x = Util.round(((float) tamanho /
47                                                1000f) * p.x, 0);
48
49                          p.y = Util.round(((float) tamanho /
50                                                1000f) * p.y, 0);
51
52                          points.add(p);
53                    }
54              }
55        }
56        return points;
57  }
```

Desenvolvimento

Implementação: Localizar pontos de interesse

- Ilustração do código
- Matriz de pixels
- Pixels “Enormes”



Tagarela

Desenvolvimento

Implementação: Evento de toque na tela

```
467  @Override
468  public boolean onTouch(View v, MotionEvent event) {
...                                    ...
488                      int aRGB = drawingCache.getPixel((int) p.x, (int) p.y);
489                      int alpha = Color.alpha(aRGB);
490                      if (aRGB != 0) {
491                            points.add(p);
492                            int remove = -1;
493                            for (int i = 0; i < wayPoints.size(); i++) {
494                                  PointF wayP = wayPoints.get(i);
495
496                                  // Testa a Bounding Box do wayPoint
497                                  if (((p.x >= wayP.x-(dimWayPoint/2)) &&
498                                        (p.x <= wayP.x+(dimWayPoint/2))) &&
499                                        ((p.y >= wayP.y-(dimWayPoint/2)) &&
500                                         (p.y <= wayP.y+(dimWayPoint/2)))) {
501                                        remove = i;
502                                  }
503                            }
504                            if (remove >= 0) {
505                                  wayPoints.remove(remove);
506                            }
```

Tagarela

Desenvolvimento

Implementação: Evento desenhar

```
375  @Override
376  protected void onDraw(Canvas canvas) {
377          super.onDraw(canvas);
378
379          if ((!simboloCarregado) && (!readOnly))
380                  return;
381
382          paint.setColor(Color.RED);
383          PointF pOld = null;
384          float x = 0;
385          float y = 0;
386
387          for (PointF p : points) {
388                  if (pOld == null)
389                          pOld = p;
390
391                  canvas.drawCircle(p.x, p.y, dimPincel, paint);
392                  x = p.x;
393                  y = p.y;
394                  pOld = p;
395          }
396
397          if (!readOnly) {
398                  if (points.size() > 0) {
399                          canvas.drawBitmap(hunter, x -
400                                  ((float) hunter.getWidth()/2f),
401                                  y - ((float) hunter.getHeight()/2f), paint);
402                  }
403
404                  for (PointF p : wayPoints) {
405                          canvas.drawBitmap(prey, p.x -
406                                  ((float) prey.getWidth()/2f),
407                                  p.y - ((float) prey.getHeight()/2f), paint);
408
409                  }
410          }
411  }
```

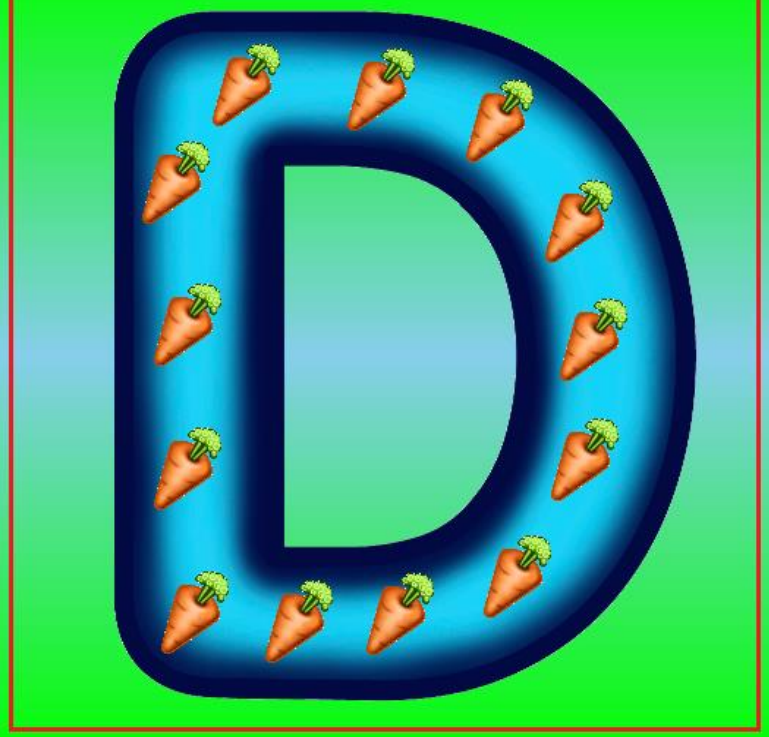
Desenvolvimento

Operacionalidade

Desenvolvimento

Operacionalidade

ABCDEFGHIJKLMNOPQRSTUVWXYZ (a)



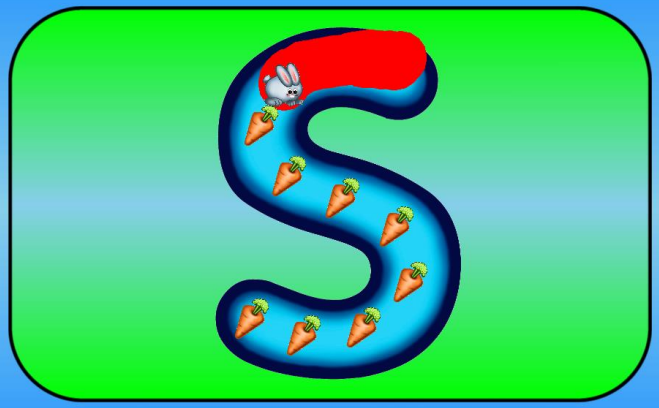
(b)

ABC (c)

Desenvolvimento

Operacionalidade

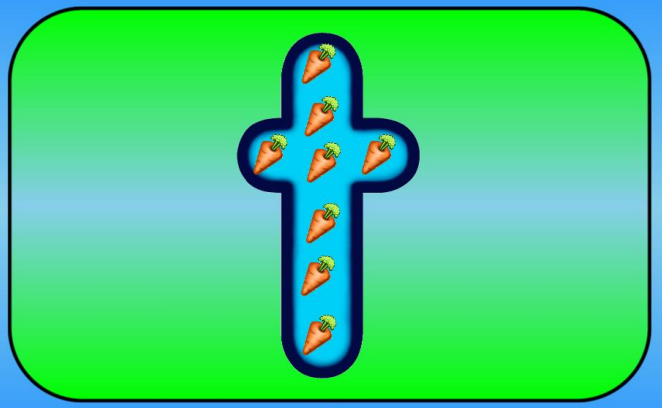
Stefanie Reetz



Stefanie Reetz

Stefanie Reetz

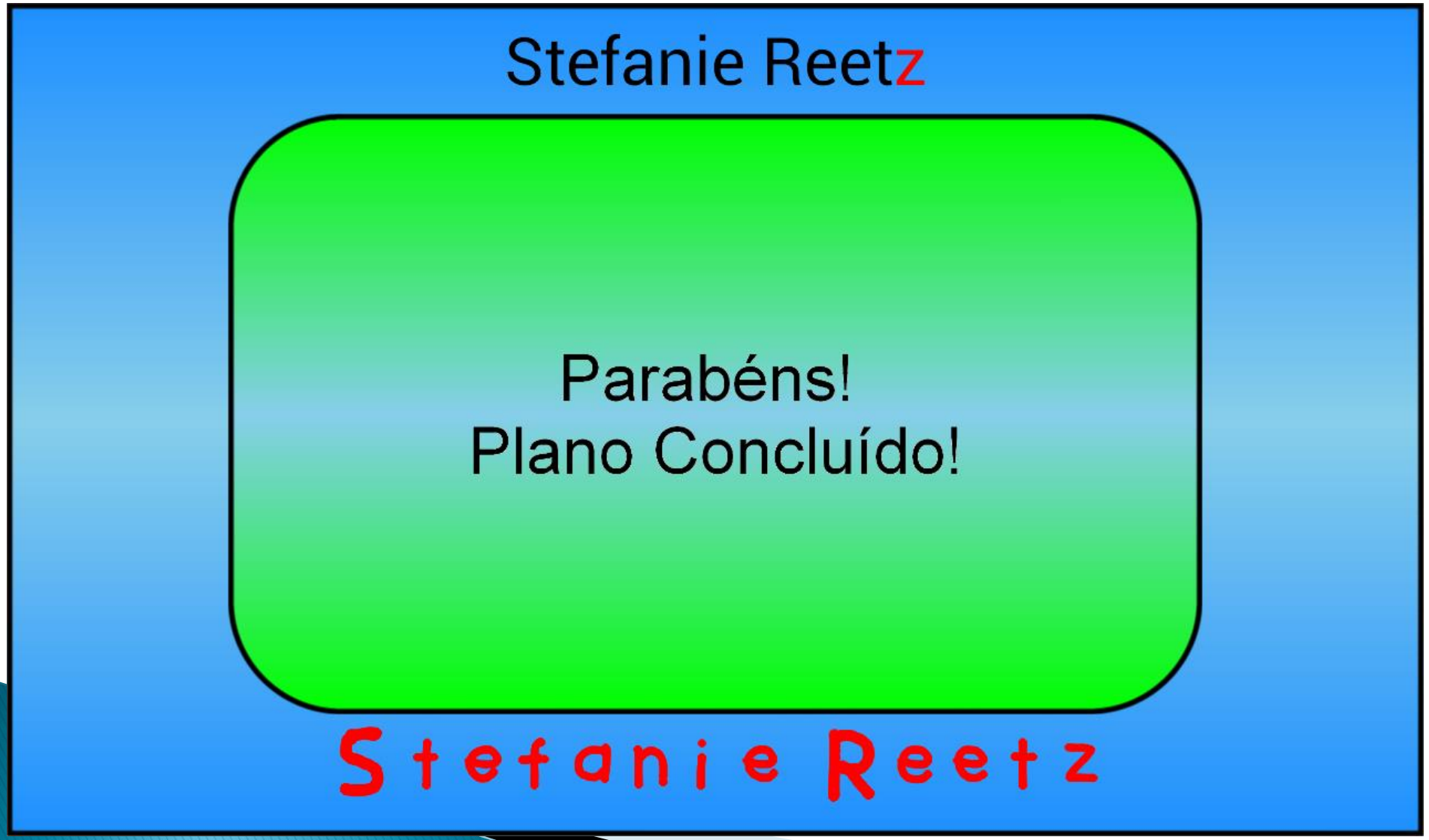
Stefanie Reetz



S

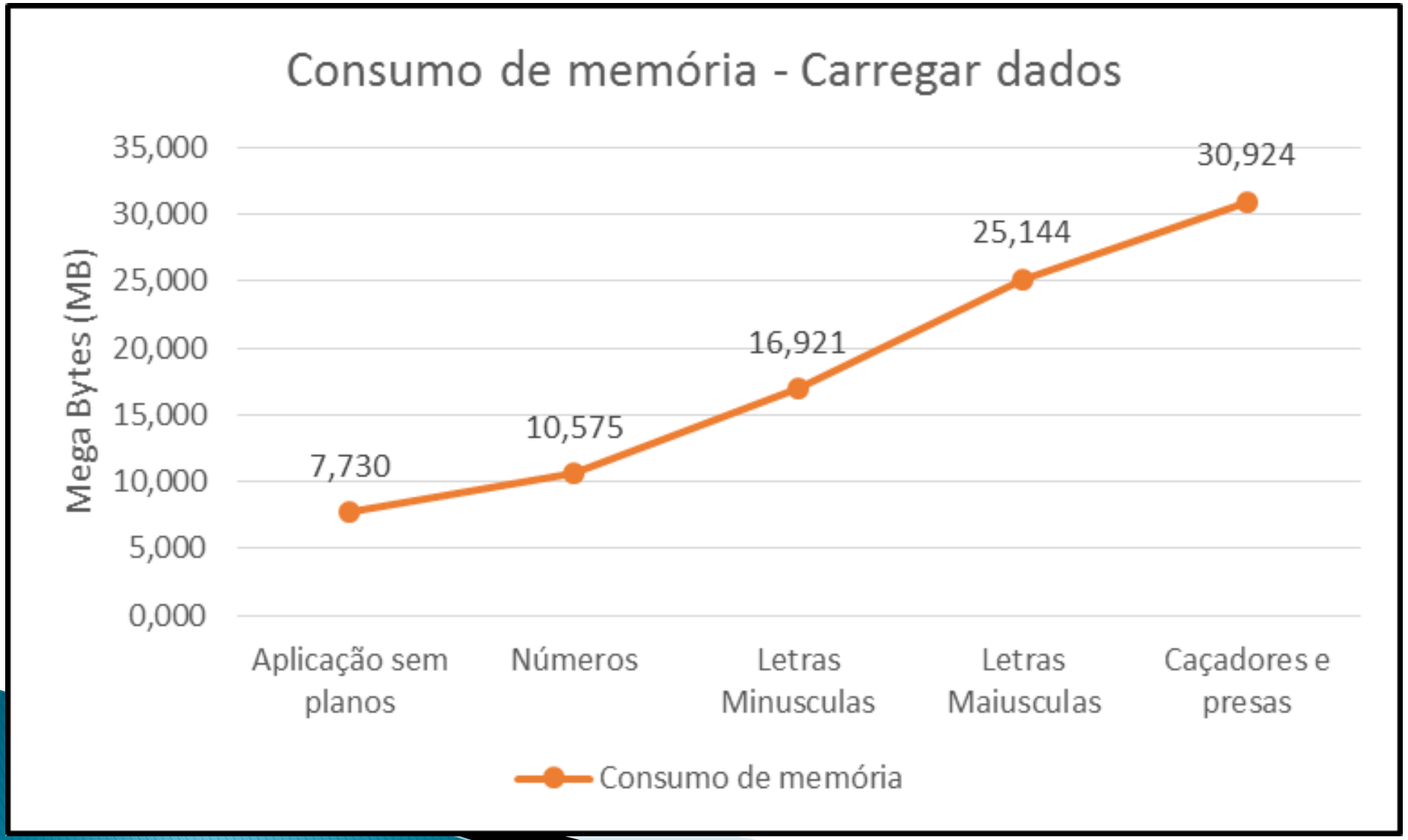
Desenvolvimento

Operacionalidade



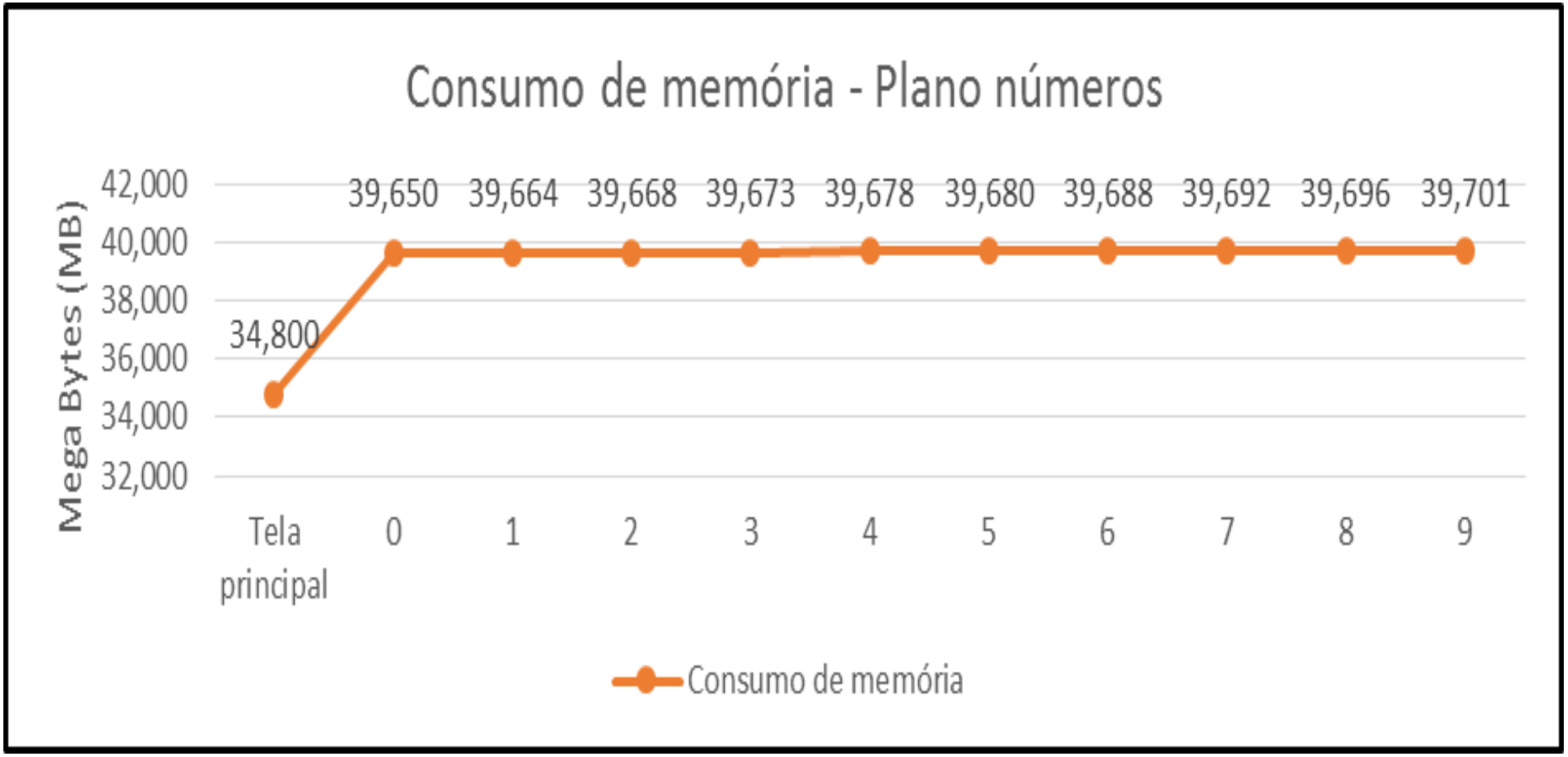
Desenvolvimento

Resultados e Discussões: Consumo de memória



Desenvolvimento

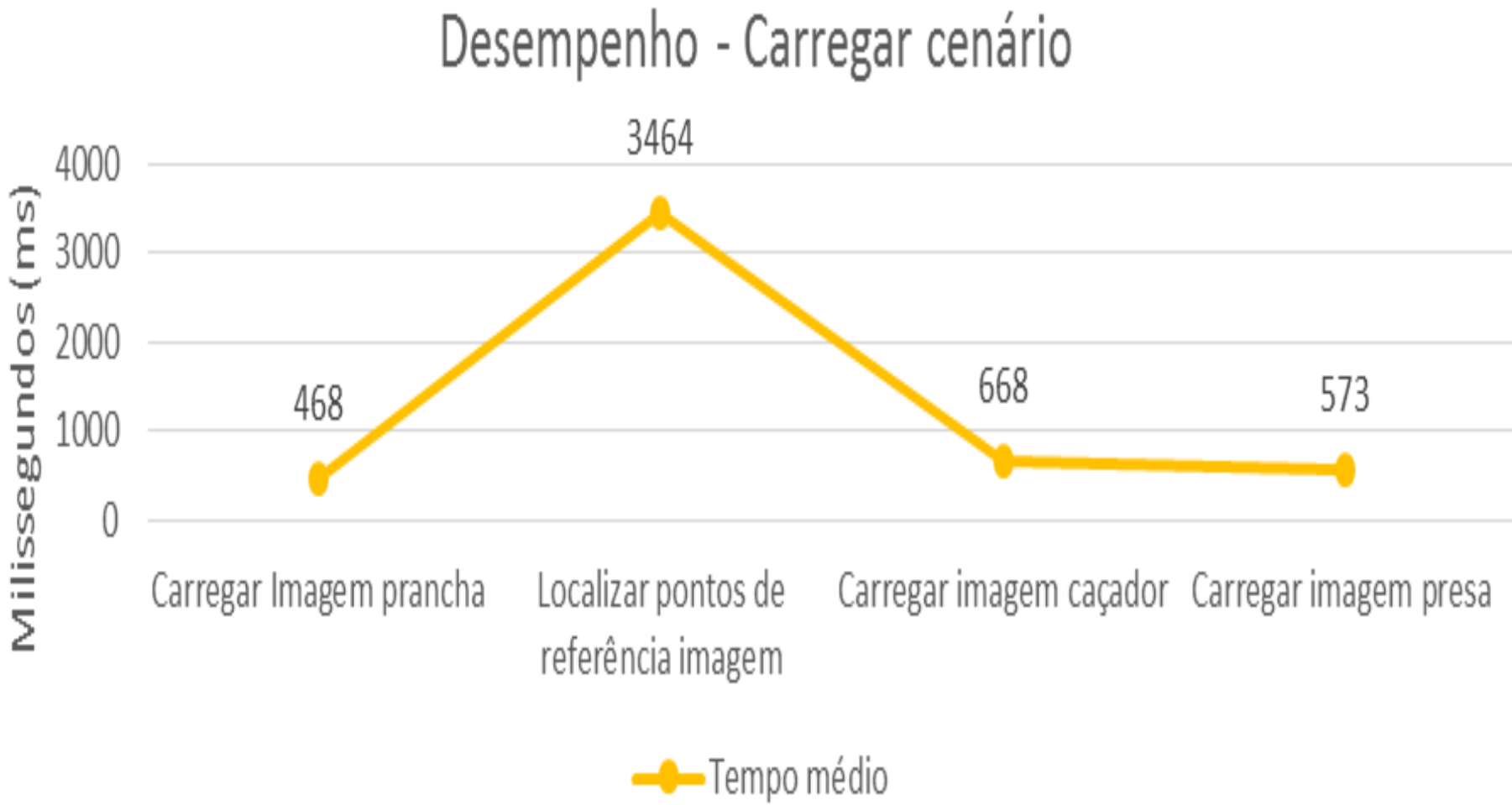
Resultados e Discussões: Consumo de memória



Tagarela

Desenvolvimento

Resultados e Discussões: Desempenho



Desenvolvimento

Resultados e discussões: Comparação com trabalhos correlatos

	Jogo 2D desenvolvido	Fabeni (2012)	Dibugrama	Desenhe e Aprenda a Escrever
Apelo pedagógico	X	X	X	X
Possui vários cenários	X	X	X	X
Possibilita criar planos/cenários	X	X		X
Utiliza áudio	X	X	X	X
Aplicativo gratuito	X	X	X	
Possui planos/cenários nativos	X		X	X
Trabalha letras e números	X			X
Possibilita acentuação nas letras				X

Desenvolvimento

Resultados e Discussões: Usabilidade

- Professora: Luscimar Rech Berkenbrock
  - Professora de Apoio Pedagógico (PAP)
  - Trabalha com uma criança com necessidades especiais do 4ª ano na Escola Básica Municipal Machado de Assis.
- Fonoaudiólogo: Rodrigo França
  - Especialista na área de comunicação alternativa

Desenvolvimento

Resultados e Discussões: Usabilidade

FURB - Universidade Regional de Blumenau
Curso de Ciência da Computação – Bacharelado
Projeto Tagarela – gcg.inf.furb.br/tagarela
TCC - Jogo de letras/números voltado para tecnologia assistiva no Android

Questionário de Avaliação

1. **Nome:** Luciana Beck Berkenbrock

2. **A utilização de imagens para representar os pontos a serem percorridos pelo aluno ajudam a estimular o aprendizado?**
   1 - [x] Sim
   2 - [ ] Não

3. **A utilização do áudio para representar o símbolo ajuda no reconhecimento da letra/número?**
   1 - [ ] Não, ajudou
   2 - [ ] Ajudou pouco
   3 - [ ] Ajudou
   4 - [x] Ajudou muito

4. **Como você avalia a utilização sequencial de pranchas durante o jogo?**
   1 - [ ] Ruim
   2 - [ ] Regular
   3 - [ ] Bom
   4 - [x] Ótimo

Desenvolvimento

Resultados e Discussões: Usabilidade

Continuação...

5. **Com base nas respostas anteriores, quais seriam os principais "Pontos Positivos" no uso do aplicativo?**

A estimulação visual do aplicativo incentiva, motiva a aluna!

6. **Com base nas respostas anteriores, quais seriam os principais "Pontos Negativos" no uso do aplicativo?**

Nada observado.

7. **Com base nas respostas anteriores, quais seriam as suas "Sugestões" de melhoria para o aplicativo?**

Os aplicativos estão bem elaborados. Acredito que uma sugestão seria elaborar aplicativos diversos, contribuindo cada vez mais para o conhecimento e crescimento da aluna.

Obs.: caso precise de espaço, favor usar o verso desta folha.

Desenvolvimento

Resultados e Discussões: Usabilidade

Tagarela

Conclusão

- Objetivos alcançados com sucesso
- Utilização de todas as letras e números da língua portuguesa
- Possibilidade de criar planos customizados
- Limitações:
  - Tempo médio para a prancha ser exibida na tela
  - Palavras ou nomes com acentuações

Extensões

- implementar um editor de pontos de referências nas imagens dos símbolos
- implementar variedade de texturas para o caminho do símbolo e possibilitar ao usuário selecioná-las
- permitir que o usuário crie planos customizados com acentuações nas letras
- analisar a possibilidade de utilizar um sintetizador de voz para reproduzir o significado da palavra ou frase dos planos customizados
- analisar a possibilidade de incluir músicas de ambiente nos cenários
- adaptar o aplicativo para todas as dimensões de tela disponível para plataforma Android
- disponibilizar o aplicativo no Play Store da Google

Demonstração prática

Fundamentação teórica

Tecnologia assistiva e jogos educacionais

- Tecnologia assistiva
  - Recursos e serviços
  - Autonomia
  - Independência
  - Qualidade de vida
  - Inclusão social

- Jogos educacionais
  - Letras/Números/Cores/Formas
  - Brincando e aprendendo

Desenvolvimento

Implementação: Evento de toque na tela

```
467  @Override
468  public boolean onTouch(View v, MotionEvent event) {
469        if (readOnly) {
470              return true;
471        }
472        // garante que evento não ultrapasse limites da view
473        if ((event.getX() > 0 && event.getX() <= getWidth()) &&
474            (event.getY() > 0 && event.getY() <= getHeight())) {
475              PointF p = new PointF();
476
477              p.x = event.getX();
478              p.y = event.getY();
479
480              edPointX.setText(String.valueOf(Util.round(p.x, 2)));
481              edPointY.setText(String.valueOf(Util.round(p.y, 2)));
482
483              if (drawingCache == null) {
484                    drawingCache = this.getDrawingCache();
485              }
486
```

Desenvolvimento

Implementação: Evento de toque na tela

- Continuação...

```
487                 if (drawingCache != null) {
488                         int aRGB = drawingCache.getPixel((int) p.x, (int) p.y);
489                         int alpha = Color.alpha(aRGB);
490                         if (aRGB != 0) {
491                                 points.add(p);
492                                 int remove = -1;
493                                 for (int i = 0; i < wayPoints.size(); i++) {
494                                         PointF wayP = wayPoints.get(i);
495
496                                         // Testa a Bounding Box do wayPoint
497                                         if (((p.x >= wayP.x-(dimWayPoint/2)) &&
498                                                 (p.x <= wayP.x+(dimWayPoint/2))) &&
499                                                 ((p.y >= wayP.y-(dimWayPoint/2)) &&
500                                                  (p.y <= wayP.y+(dimWayPoint/2)))) {
501                                                 remove = i;
502                                         }
503                                 }
504                                 if (remove >= 0) {
505                                         wayPoints.remove(remove);
506                                 }
507
508                                 invalidate();
509                                 if ((remove >= 0) && (wayPoints.size() == 0)) {
510                                         playSound();
511                                         playAnimationDestacar();
512                                 }
513                         }
514                         else
515                                 Util.vibrar(getContext(), 100);
516                 }
517         }
518         return true;
519 }
```

Desenvolvimento

Implementação: Geração do histórico

```
261 public void gerarHistorico(){
262         if (simboloView.getPoints().size() > 0) {
263 
264                 android.view.ViewGroup.LayoutParams
265                 lParams = new LayoutParams(lParamsImg);
266 
267                 char c = simboloView.getSimbolo().getSimboloBD().
268                                 getName().toCharArray()[0];
269 
270                 // Letras Maiúsculas ou Números
271                 if (((c >= 65) && ((c <= 90))) ||
272                     ((c >= 48) && (c <= 57))) {
273                         lParams.width = 70;
274                         lParams.height = 70;
275                 }
276 
277                 List<PointF> points = new ArrayList<PointF>();
278                 for (PointF pointF : simboloView.getPoints()) {
279                         pointF.x = pointF.x * ((float) lParams.width /
280                                         simboloView.getWidth());
281 
282                         pointF.y = pointF.y * ((float) lParams.height /
283                                         simboloView.getHeight());
284 
285                         points.add(pointF);
286                 }
287 
288                 gerarViewHistorico(points, lParams);
289 
290                 if ((pranchaIndex > 0) &&
291                                 (plano.getPrancha(pranchaIndex-1).
292                                 getSimbolo().getSimboloBD().getName().
293                                 equals(" "))) {
294 
295                         gerarViewHistorico(new ArrayList<PointF>(),
296                         new android.view.ViewGroup.LayoutParams(20, 50));
297                 }
298         }
299 }
```

Desenvolvimento

Implementação: Geração do histórico

```
301 @SuppressLint("NewApi")
302 public void gerarViewHistorico(List<PointF> points,
303                 android.view.ViewGroup.LayoutParams lParams){
304         SimboloView img = new SimboloView(getApplicationContext(), true);
305
306         img.setLayoutParams(new LayoutParams(lParams));
307         img.setPoints(points);
308         img.setReadOnly(true);
309         img.setOnTouchListener(null);
310         img.setVisibility(View.VISIBLE);
311
312         if (lParams.width == 70) {
313                 jogoLayoutHistorico.setGravity(Gravity.CENTER | Gravity.BOTTOM);
314         }
315         else
316                 jogoLayoutHistorico.setGravity(Gravity.CENTER);
317
318         jogoLayoutHistorico.addView(img);
319
320         ObjectAnimator
321         scaleXIn = ObjectAnimator.ofFloat(img, "scaleX", 0f, 1f);
322
323         ObjectAnimator
324         scaleYIn = ObjectAnimator.ofFloat(img, "scaleY", 0f, 1f);
325
326         ObjectAnimator
327         rotateClockWise = ObjectAnimator.ofFloat(img,"rotation", 0f, 360f);
328
329         AnimatorSet set = new AnimatorSet();
330         set.play(scaleXIn).with(rotateClockWise).with(scaleYIn);
331         set.setDuration(1000);
332         set.setStartDelay(0);
333         set.start();
334 }
```

Desenvolvimento

Implementação: Animações

```
142    @SuppressLint("NewApi")
143    public void playAnimationIn() {
144        AplicarNovoCenario();
145
146        this.clearAnimation();
147        ObjectAnimator scaleXIn = ObjectAnimator.ofFloat(this, "scaleX", 0f, 1f);
148        ObjectAnimator scaleYIn = ObjectAnimator.ofFloat(this, "scaleY", 0f, 1f);
149        ObjectAnimator rotateClockWise = ObjectAnimator.ofFloat(this,"rotation", 0f, 360f);
150
151        AnimatorSet set = new AnimatorSet();
152        set.play(scaleXIn).with(rotateClockWise).with(scaleYIn);
153        set.setDuration(1000);
154        set.setStartDelay(0);
155
...                              ...

185
186        set.start();
187    }
188
```