IT ШКОЛА SAMSUNG

Индивидуальный проект

Игра-головоломка «Cat Universe»

Город: Воронеж

Площадка: ВГУ (группа VGUP92)

Учащийся: Гладких Яна

Преподаватель: Рудин Павел Игоревич

Дата: 27 мая 2020 г.

IT ШКОЛА SAMSUNG

Введение

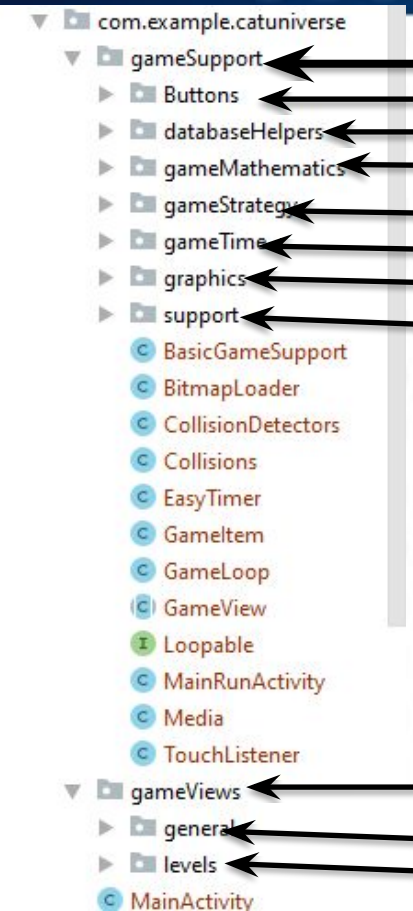
Идея приложения .
В игре представлены математические, стратегические и временные уровни. Цель временных уровней – за определенное время добраться до финиша, выполнив при этом ряд поставленных задач; Стратегических – расставить персонажей на поле так, чтобы они смогли одолеть врагов; Математических – запоминать различные формулы, теоремы в игровом виде. За прохождение уровней игрок получает награды – новых персонажей. Персонажами в игре являются коты. Полученных персонажей необходимо размещать в комнаты, тогда ими можно будет играть в дальнейшем. Таким образом, игрок имеет возможность не только получать удовольствие , проходя уровни, но и создавать собственную «кото-ферму», стремясь когда-нибудь заполучить всех возможных котов, что не так просто!

Актуальность, практическое применение:
Уровни на время помогают повысить реакцию, умение быстро находить выход из какой-либо ситуации. Стратегические помогают строить в голове сложные расчёты, продумывать шаги, мыслить аналитически. Математические, в свою очередь, помогут школьникам 5-11 классов освоить теорию математики. Ну и ,несомненно , важнейшая цель любой игры – получение удовольствия.

разработки Android Studio.

IT ШКОЛА SAMSUNG

Структура проекта



Компоненты, поддерживающие игру

Кнопки

Вспомогательные классы для бд

Поддержка математических уровней

Поддержка стратегических уровней

Поддержка уровней на время

Поддержка графики

Вспомогательные классы для GameView

Классы, наследующие GameView

Основные элементы меню игры, выбор уровней, переход в комнаты

Классы, объединяющие все уровни одноного типа

IT ШКОЛА SAMSUNG

Игровой движок

```
public class GameLoop extends SurfaceView implements Runnable {
    private boolean running = false;
    Thread gameThread = null;
    MainRun mainRun;
    Bitmap bmp;
    SurfaceHolder surfaceHolder;
    Canvas canvas;
    Rect rect;
```

Переменные, определенные в классе

```
@Override
public void run() {
    float beginTime = System.nanoTime();
    float delta = 0;
    while (running) {
        int UPDATE_TIME = 1000000000 / 60;
        delta += (System.nanoTime() - beginTime) / UPDATE_TIME;
        beginTime = System.nanoTime();
        if (delta > 1) {
            runGame();
            delta--;
        }
```

Перейдём к самой низкоуровневой части программы . Логика работы игры, перерисовки картинок, обновления экрана реализована в классе **GameLoop**.

GameLoop наследует класс SurfaceView для рисования кадров в фоновых потоках и имплементирует интерфейс Runnable, который служит для запуска потока.

Метод run(), переопределенный с помощью интерфейса Runnable производит обновление игры 60 раз за одну секунду. Для этого занесем в переменную beginTime время в нано секундах , полученное в данный момент. В переменную delta будет заноситься разница между настоящим моментом и beginTimr, деленная на время обновления.

Если delta больше 1, тогда уменьшим её до нуля. Этот процесс будет повторяться бесконечно в течение всего игрового процесса.

IT ШКОЛА SAMSUNG

Перерисовка

```
private void runGame() {
    if (surfaceHolder.getSurface().isValid()) {
        canvas = surfaceHolder.lockCanvas();
        canvas.getClipBounds(rect);
        canvas.drawBitmap(bmp,  src: null, rect,  paint: null);
        mainRun.getCurrentView().run();

        surfaceHolder.unlockCanvasAndPost(canvas);
    }
}

public void startGame() {
    if (running) return;
    running = true;
    gameThread = new Thread( target: this);
    gameThread.start();
}
```

Для отрисовки используется класс Canvas. Переменной canvas в методе runGame() присваивается элемент класса Canvas, полученный из объекта surfaceHolder. После чего строкой mainRun().getCurrentView().run() вызывается текущий игровой кадр, получаемый из mainRun, о котором я расскажу чуть позже. В конце surfaceHolder должен разблокировать canvas.

Сам метод runGame() запускается в переопределенном методе run() каждый раз, когда delta>0 т.е. 60 раз в секунду. Именно здесь происходит обновление игрового кадра.

Метод startGame() запускает GameLoop.

IT ШКОЛА SAMSUNG

Поддержка графики

Класс **GamePaint** отвечает за графику игры, предоставляет методы для работы с ней.

```
public class GamePaint {

    private AssetManager am;
    private Bitmap mainBitmap;
    private Canvas canvas;
    private Paint paint;

    public GamePaint(AssetManager am, Bitmap mainBitmap) {
        this.am = am;
        this.mainBitmap = mainBitmap;
        this.canvas = new Canvas(mainBitmap);
        this.paint = new Paint();
        // Игровой шрифт
        Typeface mainFont = Typeface.createFromAsset(am,  path: "try3.otf");
        paint.setTypeface(mainFont);
        paint.setAntiAlias(true);
    }
```

Конструктор принимает объект AssetManager, который будет получен из context главной activity и Bitmap.
В классе также определены переменные Paint , с помощью которого можно будет задать цвет, размер шрифта и т.д. Typeface подключает в игру единый шрифт.

В классе также определен ряд методов для работы с графикой – загрузка изображений, вывод текста на экран и т.д.

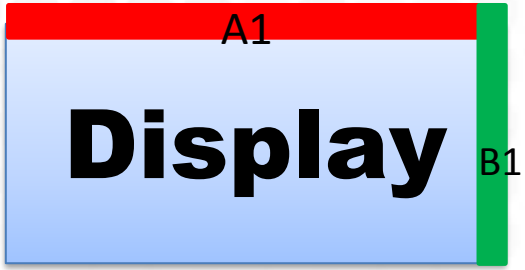
```
class Media {
    private AssetManager assetManager;

    Media(Activity activity) {
        activity.setVolumeControlStream(AudioManager.STREAM_MUSIC);
        assetManager = activity.getAssets();
    }

    Music setMusic(String fileName) throws IOException {
        return new Music(assetManager.openFd(fileName));

    }

    public class Music implements MediaPlayer.OnCompletionListener {
        MediaPlayer mediaPlayer;
        boolean isPrepared;
```

Класс **Media** служит для подключения музыки. Аналогично GamePaint потребуется получить экземпляр класса AssetManager из главной activity. Media имеет вложенный класс Music, который реализует интерфейс MediaPlayer.OnComletionListener.

Настройка обработки событий относительно размера экрана телефона

Заглянем в **MainRunActivity**– основной класс, который наследует AppCompatActivity. В методе onCreate() создан экземпляр класса Point, который содержит в себе 2 координаты x и y. Теперь нужно получить размер дисплея телефона, на котором запущено приложение и поместить это значение в point.
Далее заданы стандартная высота и ширина экрана. Теперь этими размерами определяется тот самый основной Bitmap, на котором и будет происходить отображение игры. Его размер всегда будет определенным – 800*600. В дальнейшем для улучшения качества изображения можно изменить его. Переменным w и h присваиваются отношения ширины дисплея к стандартной ширине Bitmap'a и длины дисплея к стандартной длине Bitmap'a соответственно.

```
Point point = new Point();
getWindowManager().getDefaultDisplay().getSize(point);

Bitmap bmp = Bitmap.createBitmap( width: 800,  height: 600, Bitmap.Config.ARGB_8888);

float w = (float)800 / point.x;
float h = (float)600 / point.y;

if (w == h) touchListener = new TouchListener(gameLoop, w);
else touchListener = new TouchListener(gameLoop, w, h);
```



Если дисплей смартфона подобен стандартному размеру изображения, то есть

$$\frac{A2}{A1} = \frac{B2}{B1} = k$$

В этом случае, TouchListener'у передается любая переменная(w или h). В противном случае, для соотнесения высот и ширин будут использоваться их отношения.

IT ШКОЛА SAMSUNG

Обработка событий

Главное назначение этих отношений/коэффициента подобия экранов – подстроить слушатель событийпод размер изображения. Таким образом, приложением можно будет пользоваться на как на самом маленьком смартфоне, так и на планшете с большим экраном.

Прейдём в класс **TouchListener**

```
public class TouchListener implements View.OnTouchListener {

    private float touchX, touchY;
    private boolean touchDown, touchUp;
    private boolean swipe, swipeUp, swipeDown;
    private float downX, downY;
    private int swipeDistance;
    private float screenWidth = 1, screenHeight = 1, k = 1;

    TouchListener(View view, float screenWidth, float screenHeight) {
        view.setOnTouchListener(this);
        this.screenWidth = screenWidth;
        this.screenHeight = screenHeight;
        swipeDistance = 50;
    }

    TouchListener(View view, float k) {
        view.setOnTouchListener(this);
        this.k = k;
        swipeDistance = 50;
    }
```

Координаты прикосновений

Вернут true при удержании/ отрыве пальца от экрана

Свайпы

Координаты при удержании

Запрашиваемый размер свайпа

Конструктор, который используется если экраны не подобны

Конструктор, который используется если экраны подобны

IT ШКОЛА SAMSUNG

Обработка событий

```
@Override
public boolean onTouch(View view, MotionEvent event) {

    synchronized (this) {
        touchDown = false;
        touchUp = false;

case MotionEvent.ACTION_UP:
    touchX = event.getX() / screenWidth * k;
    touchY = event.getY() / screenHeight * k;


float upX = event.getX();
float upY = event.getY();

float deltaX = downX - upX;
float deltaYUp = downY - upY;
float deltaYDown = upY - downY;

if (Math.abs(deltaYUp) > swipeDistance) {
    if (deltaYUp > 0) swipe = true;
}
```

В переопределенном методе onTouch () реализованы типы касаний – удержание, отрыв пальца и свайпы(свайп вверх , свайп вниз , обобщенный свайп).

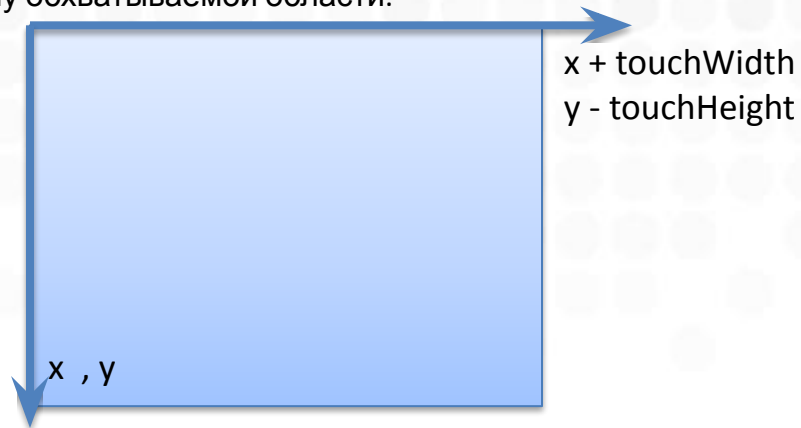
Данный пример демонстрирует обработку отрыва пальца от экрана. Переменные touchX и touchY получают x/y-координаты касания, умноженные либо на отношение длин сторон Bitmap'а к дисплею, либо на коэффициент подобия.

Расчёт свайпов (провести пальцем по экрану определенное расстояние). В переменные deltaX и deltaY заносится разница между координатой последнего удержания пальца и его отрыва от экрана. Если это расстояние удовлетворяет заданному расстоянию (я задаю размер по y 50 относительно экрана), то переменной swipe присваивается значение true

Обработка событий

```
public boolean down(int x, int y, int touchWidth, int touchHeight) {
    if (touchDown)
        if (touchLimits(x, y, touchWidth, touchHeight)) {
            touchDown = false;
            return true;
        }
    return false;
}

private boolean touchLimits(int x, int y, int touchWidth, int touchHeight) {
    return touchX >= x && touchX <= x + touchWidth && touchY <= y && touchY >= y - touchHeight;
}
```

Метод touchLimits определяет, находится ли касание в заданном интервале и возвращает true/false. Для этого он принимает начальные точки – x и y, высоту и ширину обхватываемой области.



IT ШКОЛА SAMSUNG

Игровые кадры

```
lic abstract class GameView {

 private MainRunActivity mainRunActivity;
 private GamePaint gamePaint;

 public static int screenWidth;
 public static int screenHeight;
 protected int stars;

 public GameView(MainRunActivity mainRunActivity) {
     this.mainRunActivity = mainRunActivity;
     screenWidth = mainRunActivity.getGamePaint().getMainBitmap().ge
     screenHeight = mainRunActivity.getGamePaint().getMainBitmap().g
     gamePaint = mainRunActivity.getGamePaint();
 }

 public abstract void run();

 public abstract void repaint();

 int getStars() { return stars; }

 public MainRunActivity getMainRunActivity() { return mainRunActivit

 public GamePaint getGamePaint() { return gamePaint; }
```

Абстрактный класс GameView отвечает за отображение кадра игры. В нём представлено 2 абстрактных метода – run() и repaint().

Метод run() – тот самый метод, который GameLoop вызывает 60 раз в секунду в методе runGame(). Метод repaint() – вспомогательный. Как правило, в нём описывается изменение параметров объектов. Вызывается он внутри метода run(), перед выполнением основного кода.

Статические переменные screenWidth/Height помогают получить размеры Bitmap'а.Т.к. GameView зачастую реализует в себе уровни игры, в нём определена переменная stars, которая возвращает кол-во звёзд, полученных за прохождение уровня.

Метод runGame в классе GameLoop

```
private void runGame() {
    if (surfaceHolder.getSurface().isValid()) {
        canvas = surfaceHolder.lockCanvas();
        canvas.getClipBounds(rect);
        canvas.drawBitmap(bmp,  src: null, rect,  paint: null);
        mainRun.getCurrentView().run();
```

IT ШКОЛА SAMSUNG

Объединение компонентов движка в единую сущность

Теперь переместимся в **MainRunActivity** – основной класс, который поддерживает работу всей игры. MainRunActivity наследует AppCompatActivity и является активностью.
В методе onCreate(), который вызывается при запуске активности и задает все начальные параметры, инициализируются основные составляющие движка.

1. **GamePaint** служит для поддержки графики и работы с ресурсами. В качестве параметров требуется передать AssetManager, который можно получить с помощью метода activity.getAssets() и Bitmap, на котором будет отображаться игра.
2. Класс **Media** позволяют подключать в игру музыку. В качестве параметров требуется передать активность т.е. MainRun (this).
3. Класс **Point и соотношения**, как уже было сказано, служат для того, чтобы обработка касаний соответствовала размеру игрового поля.
4. **TouchListener** подключает к активности обработчик событий, умножая свои размеры на полученные соотн
5. **GameView** служит для отображения игрового кадра.

В метод setContentView(), задающий активности пользовательский интерфейс, передается **GameLoop**. Теперь с
будет отображать текущее окно активности , вызывая метод run() класса GameView 60 раз в секунду, о чём бы
сказано раннее.MainRun также указана в AndroidManifest.xml как активность. В качестве главной активности ук
класс MainActivity, наследующий MainRunActivity

```
<activity
    android:name="com.example.catuniverse.gameSupport.MainRunActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
    </intent-filter>
</activity>
```

- BasicGameSupport
- BitmapLoader
- CollisionDetectors
- Collisions
- EasyTimer
- GameItem
- GameLoop
- GameView
- Loopable
- MainRunActivity
- Media
- TouchListener

IT ШКОЛА SAMSUNG

Класс MainRunAvtivity

```
public class MainRunActivity extends AppCompatActivity {

    private GamePaint gamePaint;
    private TouchListener touchListener;
    private GameView gameView;
    private Media media;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

        Point point = new Point();
        getWindowManager().getDefaultDisplay().getSize(point);

        Bitmap bmp = Bitmap.createBitmap( width: 800,  height: 600, Bitmap.Config.ARGB_8888);

        float w = (float)800 / point.x;
        float h = (float)600 / point.y;

        media = new Media( activity: this);

        GameLoop gameLoop = new GameLoop( mainRunActivity: this, bmp);
        gamePaint = new GamePaint(getAssets(), bmp);

        if (w == h) touchListener = new TouchListener(gameLoop, w);
        else touchListener = new TouchListener(gameLoop, w, h);

        gameView = getNewView();

        setContentView(gameLoop);
        gameLoop.startGame();
    }
```

IT ШКОЛА SAMSUNG

Взаимодействие между объектами

Перейдём к объектно-ориентированной части приложения.

Рассмотрим класс **GameItem.** Он представляет собой краткое описание параметров всех объектов игры.

```
public class GameItem implements Loopable {
    protected int x, y, controlY;
    protected double speed, jumpingSpeed, collLength;
    protected Rect collisionRect;
    protected Bitmap bitmap, bitmapClicked;
```

x и *y* - координаты, отвечающие за размещение на экране.

controlY – базовая y-координата, относительно которой фон будет изменяться при передвижении игрока.

speed, *jumpingSpeed* – скорость движения/прыжка игрока.

bitmap, bitmapClicked – отображение игровых объектов. bitmapClicked используется если объект поддерживает обработку событий.

collisionRect, collLength – переменные для определения коллизий (столкновений игрока с предметами). Об их предназначении будет рассказано позже.

Для всех переменных реализованы геттеры/сеттеры.

Представленные параметры характерны практически для всех объектов в игре.

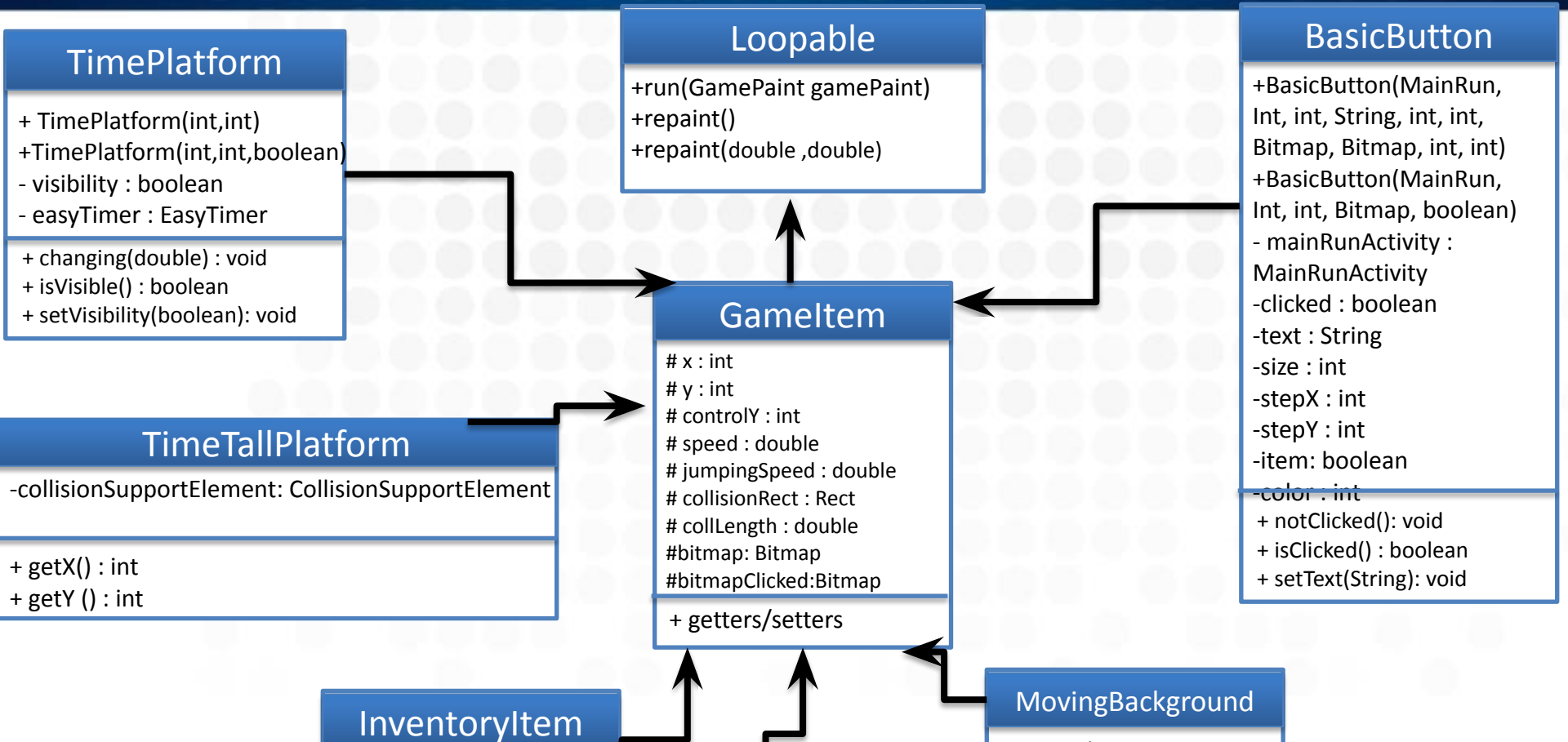
Стоит отметить, что **GameItem** имплементирует интерфейс **Loopable**, который содержит всего 3 метода - run() и две вариации repaint().

```
public interface Loopable {
    void run(GamePaint gamePaint);
    void repaint();
    void repaint(double speed, double jumSpeed)
}
```

Рассмотрим часть UML- диаграммы

IT ШКОЛА SAMSUNG

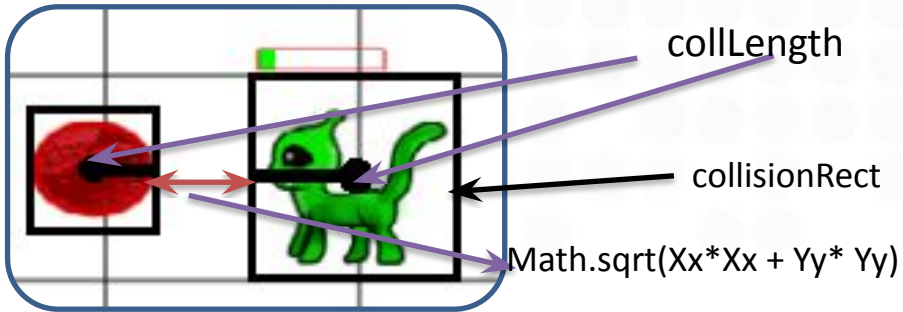
Uml-диаграмма классов, наследующих GameItem



IT ШКОЛА SAMSUNG

Обработка столкновений. Колизии

```
tic boolean checkGameItemCollision(GameItem item1, GameItem item2) {
 if (item1 != null && item2 != null) {

     double Xx = itemX(item1, item2);
     double Yy = itemY(item1, item2);

     if (Xx != 0 && Yy != 0) {
         return Math.sqrt(Xx * Xx + Yy * Yy) < item1.getCollLength() + item2.getCollLengt
     }
 }
 return false;

private static double itemX(GameItem item1, GameItem item2){
    if(item1==null||item2==null){
        return 0.0;
    }
   return item1.getCollisionRect().centerX() - item2.getCollisionRect().centerX();
}
```



Рассмотрим основной способ обработки соприкосновений в игре. Он реализован в методе checkGameItemCollision() и принимает в качестве параметров 2 объекта типа **GameItem**.

Расстояние между объектами, которое рассчитывается по формуле

$$AB = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

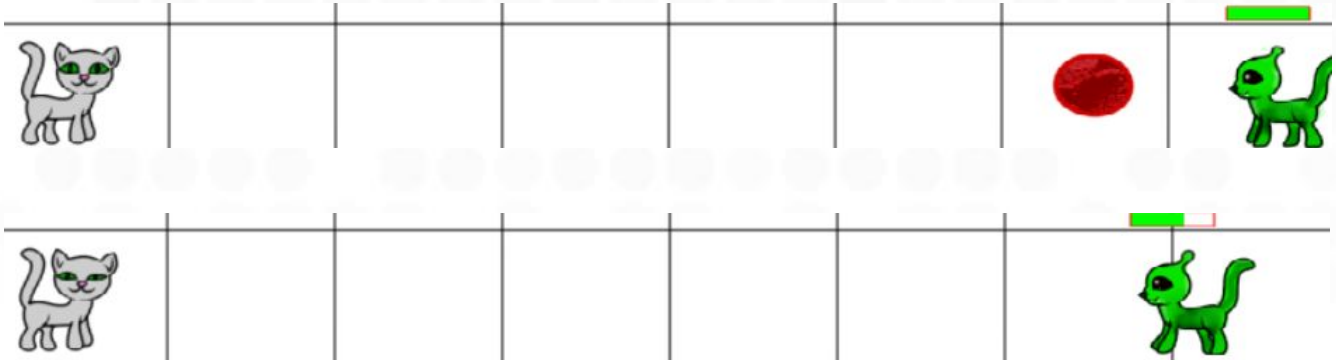
Если расстояние меньше суммы длин столкновения двух объектов, метод вернет true. Так как объекты различны по размеру, collLength иногда приходится делить в 2 раза, как показано на картинке или более. Таким образом, можно регулировать на каком расстоянии должно произойти столкновение.

Обработка столкновений. Коллизии

```
public class CollisionDetectors {
    public static boolean checkTwoItemCollision(GameItem gameItem, GameItem gameItem2) {
        return checkGameItemCollision(gameItem, gameItem2);
    }
    public static boolean checkPlayerItemCollision(GameItem gameItem) {
        return checkGameItemCollision(timePlayer, gameItem);
    }
```

В классе **CollisionDetectors** объявлены все методы, отвечающие за обработку столкновений. В большинстве случаев используется представленный способ, но для некоторых объектов пришлось создать особенные обработчики.

Стоит отметить, что этот метод применим для всех объектов, потому что он принимает в качестве параметров не экземпляр конкретного класса, а экземпляр любого класса, наследующего **GameItem**. Именно такой подход помогает сократить лишний код.



IT ШКОЛА SAMSUNG

Загрузка ресурсов

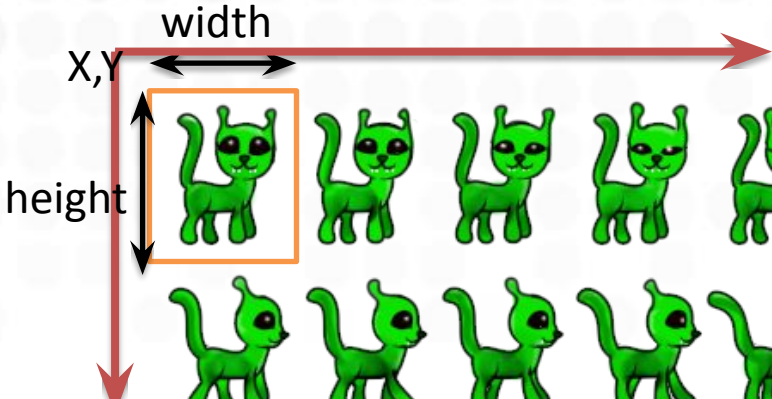
Загрузка ресурсов производится при помощи класса **BitmapLoader**. Сами ресурсы хранятся в папке assets. При загрузке картинки преобразуются в Bitmap, файлы .mp3 в Media.Music

```
public class BitmapLoader {
    public static ArrayList<ImageSet> imageSets = new ArrayList<>();

    //Музыка
    public static Media.Music music, menuMusic;

    //коты
    private static Bitmap grayCat, grayCatReversed, orangeCat, orangeCatReversed, shadowCat,
            shadowCatReversed, greenAlienCat, greenAlienCatReversed;

//Загрузка спрайтов котов
grayCatReversed = gamePaint.createNewGraphicsBitmap( name: "grayCatReversed.
grayCat = gamePaint.createNewGraphicsBitmap( name: "grayCat.png");
orangeCat = gamePaint.createNewGraphicsBitmap( name: "orangeCat.png");


private void createRocketList(ArrayList<Bitmap> rocketRight, Bitmap bmpRegular) {
    int count = 277;

    for (int i = 0; i < 3; i++) {
        rocketRight.add(Bitmap.createBitmap(bmpRegular, count,  y: 282,  width: 110,  height: 94));
        count += 110;
    }
}
```

Все переменные статические.

Загрузка изображений производится при помощи метода createNewGraphicsBitmap, который представлен в классе GamePaint.

Спрайты загружаются с помощью метода Bitmap.createBitmap из стандартной библиотеки android и записываются в ArrayList'ы.

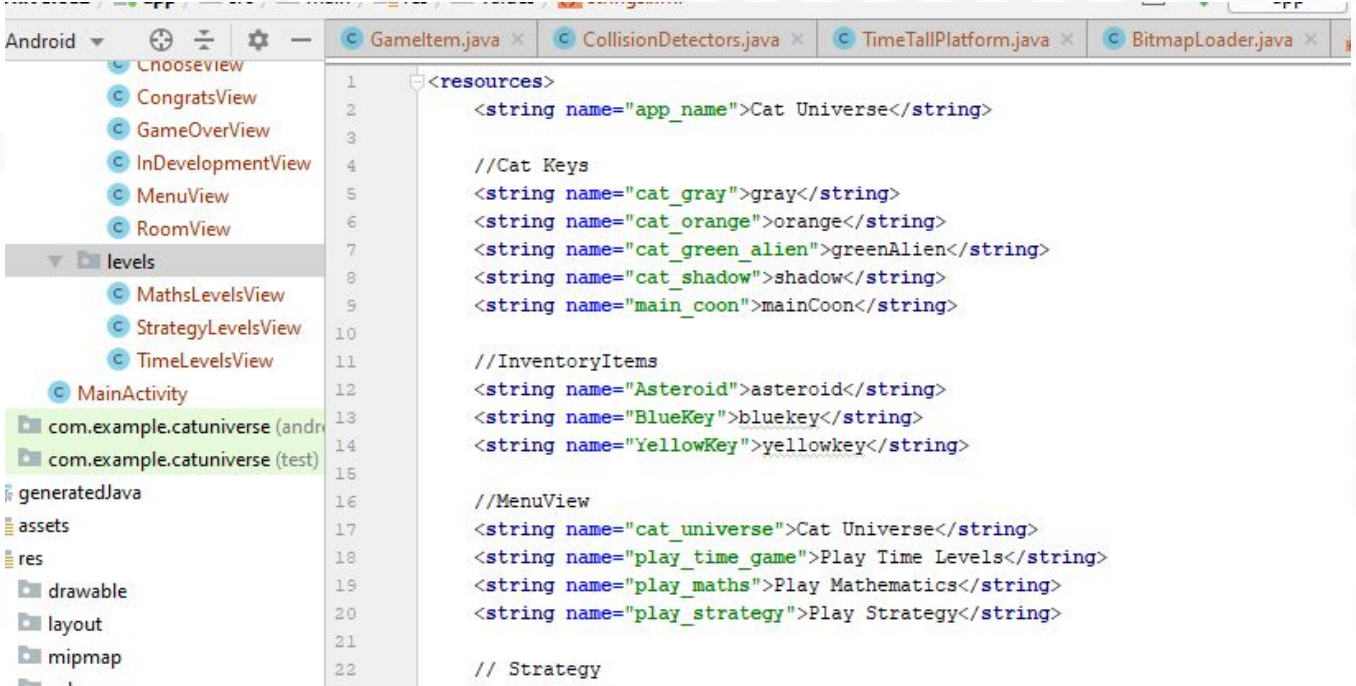


IT ШКОЛА SAMSUNG

Строковые значения

Все строковые значения хранятся в директории values в xml-файле strings.xml. Доступ к ним можно получить через главную активность.

```
super.getGamePaint().write("Congratulations!",  x: 230,  y: 50, color,  sizeText: 45);
super.getGamePaint().write( text: super.getMainRunActivity().getString(R.string.you_completed_level)
```



IT ШКОЛА SAMSUNG

Спрайтовая анимация

Для того, чтобы игроки/враги и т.д. «двигались» используется спрайтовая анимация. Кадры быстро сменяются и создаётся иллюзия движения. Всё это реализовано в классе **SpriteAnimation**.

```
public class SpriteAnimation {
    private Bitmap sprite, sprite1, sprite2, sprite3, sprite4, sprite5, sprit
    private EasyTimer easyTimer;
    private byte control, savedControl;
    private int del;

    //Конструктор класса. Принимает 1 параметр - ArrayList Bitmap'ов
    public SpriteAnimation(@NotNull ArrayList<Bitmap> spriteList) {

        //Если размер ArrayList'а позволяет, записываем в соответствующую пер
        if (spriteList.size() >= 1) this.sprite1 = spriteList.get(0);
        if (spriteList.size() >= 2) this.sprite2 = spriteList.get(1);
```

```
public void repaint(double delay) {
    easyTimer.startTimer();
    del++;
    if (del > delay) {
        if (control == 1) sprite = sprite1;
        if (control == 2) sprite = sprite2;
        if (control == 3) sprite = sprite3;
        if (control == 4) sprite = sprite4;
        if (control == 5) sprite = sprite5;
        if (control == 6) sprite = sprite6;
```

Конструктор класса принимает всего один параметр – ArrayList с кадрами.

Переменной control присваивается размер полученного списка (от 1 до 8).

Метод repaint() заменяет основной кадр sprite на кадр из списка, индекс которого соответствует control.

Метод out() отображает основной кадр в соответствии с координатами.

IT ШКОЛА SAMSUNG

Кнопки для универсального использования

Я не буду рассказывать о всех классах, наследующих GameItem, но класс BasicButton стоит заметить, так как это один из ...сов в игре.

```
public class BasicButton extends GameItem {
    private MainRunActivity mainRunActivity;
    private boolean clicked;
    private String text = null;
    private int size = 0;
    private int stepX = 0, stepY = 0;
    private boolean item;
    private int color;
//Конструктор кнопки без текста
public BasicButton(MainRun mainRun, int x, int y, Bitmap button, Bitmap clickedButton, boolean item) {
//Конструктор кнопки с текстом
public BasicButton(MainRun mainRun, int x, int y, @Nullable String text, int color, int size, Bitmap button, B
public void repaint() {
    if (mainRun.getTouchListener().up(x,  y: y + button.getHeight(), button.getWidth(), button.getHeight())) {
        clicked = !clicked;
```

Класс BasicButton имеет 2 конструктора : для кнопок с текстом и без текста.

Кнопка без текста

Кнопка с текстом

Обработчик нажатий на кнопку

Конструктор для кнопок без текста принимает в себя координаты x, y и изображения нажатого и не нажатого сосстояние кнопки, для кнопок с текстом добавляются параметры теста и его стиля.

Конструктор кнопки без текста имеет ещё один интересный булевый параметр item. Он сообщает классу является ли данная кнопка просто частью интерфейса (для перехода с одного кадра на другой и т.д.) или же игровым объектом. Так, дверь, которую необходимо открыть для прохождения уровня на время представляет собой BasicButton. Помимо возможности к обработке касаний, дверь перемещается по экрану вместе с фоном т.е. является часть какого-либо уровня. Поэтому BasicButton может найти множество вариантов применения . Например, игрок нашел сундук бонусами во время уровня на время, какие у него должны быть параметры? Обработка касаний – возможность открыть его при нажатии. Зачем создавать отдельные классы для подобных объектов, если BasicButton способен выполнить их задачи в одиночку.

IT ШКОЛА SAMSUNG

Сохранение информации об игре в бд SQLite

Для начала рассмотрим класс **MainActivity**. Он наследует MainRunActivity, который , в свою очередь, наследует AppCompatActivity, задает приложению пользовательский интерфейс.
Для работы с базами данных в MainActivity переопределен метод onResume(), который вызывается сразу после запуска активности.
Чтобы все данные в игре сохранялись после выхода (звезды за уровни, доступные коты и т.д.), созданы 4 сущности: timeDB, catsDB, strategyDB, mathsDB.

```
public class MainActivity extends MainRunActivity {
    public static SQLiteDatabase timeDB, catsDB, strategyDB, mathsDB;
    public static Cursor cursor, catCursor, strategyCursor, mathsCursor;
    public static String DB_PATH1, DB_PATH2, DB_PATH3;
    public static ArrayList<Level> timeLevels, strategyLevels, mathsLevels;
    public static ArrayList<Cat> listOfCats;
    public static ArrayList<CatPet> listOfPets;
```

Создание одной из таблиц:

```
@Override
protected void onResume() {
    super.onResume();
    timeLevels = new ArrayList<>();
    DB_PATH1 = this.getFilesDir().getPath() + "time.db";
    timeDB = getBaseContext().openOrCreateDatabase( s: "time.db", MODE_PRIVATE,  cursorFactory: null);
    //   timeDB.execSQL("DROP TABLE IF EXISTS levelStats");
    timeDB.execSQL("CREATE TABLE IF NOT EXISTS time (_id INTEGER, stars INTEGER)");
    timeDB.execSQL("INSERT into time (_id, stars) VALUES (1,0)");
    timeDB.execSQL("INSERT into time (_id, stars) VALUES (2,0)");
```

Самая простая база данных «levelStats» содержит информацию об уровнях на время. Она включает в себя 2 поля : _id и stars целочисленного типа.

Первое поле отвечает за нумерацию уровней, второе – за сохранение кол-ва звёзд. Изначально в поле stars заносится 0.

_id	stars
1	0

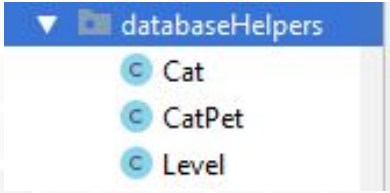
Аналогично созданы и прочие таблицы.

```
catsDB.execSQL("CREATE TABLE IF NOT EXISTS cats (_id INTEGER, name TEXT, imageSet TEXT, power INTEGER, speed
catsDB.execSQL("INSERT into cats (_id, name, imageSet, power, speed , delay , chosen, unlocked, room) VALUES
```

IT ШКОЛА SAMSUNG

Манипуляция данными

Было бы тяжело каждый раз перебирать базу данных при помощи Cursor, поэтому созданы вспомогательные классы, объединенные в папке **databaseHelpers**. Вернемся к базе с информацией о временных уровнях.



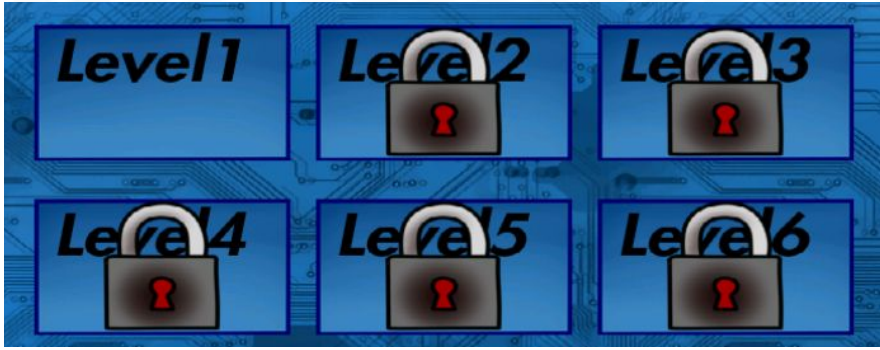

```
for (int i = 0; i < BasicGameSupport.levelsCount; i++) {
    cursor = timeDB.rawQuery( sql: "SELECT * from time WHERE _id = " + (i + 1),  selectionArgs
    if (cursor != null && cursor.moveToFirst()) {
        timeLevels.add(new Level(cursor.getInt( i: 0), cursor.getInt( i: 1)));
```

После создания бд при помощи Cursor выберем значения всех полей в текущей строке. В ArrayList <Level>timeLevels добавится новый экземпляр вспомогательного класса Level, который в качестве параметров принимает в себя значения обоих полей базы

```
public class Level {
    private int number, stars;

    public Level(int number, int stars) {
        this.number = number;
        this.stars = stars;
    }
```

Теперь манипулировать информацией об уровнях просто.
Ниже представлен отрывок кода из класса **LevelChoice**, в котором можно выбирать уровень, получая соответствующую информацию из списков.

```
switch (key) {
    case "time":
        levelButtons.get(i).repaint(MainActivity.timeLevels.get(i).getStars(), S
        break;
    case "strategy":
        levelButtons.get(i).repaint(MainActivity.strategyLevels.get(i).getStars(
        break;
    case "maths":
        levelButtons.get(i).repaint(MainActivity.mathsLevels.get(i).getStars(),
        break;
```



IT ШКОЛА SAMSUNG

Обновление данных

Так как значения полей баз данных постоянно меняются, списки тоже следует обновлять. Для этого создан ряд методов. Ниже представлен метод, обновляющий списки уровней на время.

```
private static void updateTimeDBHelpers() {
    timeLevels.clear();
    for (int i = 0; i < levelsCount; i++) {
        cursor = timeDB.rawQuery( sql: "SELECT * from levelStats WHERE _id = " + (i + 1),  selectionArgs: null);
        if (cursor != null && cursor.moveToFirst())
            timeLevels.add(new Level(cursor.getInt( i: 0), cursor.getInt( i: 1)));
    }
```

После прохождения уровня/получения нового персонажа и т.д. необходимо обновить поля базы данных.
Метод updateTimeStars обновляет количество звезд в уровнях на время. В экземпляр класса ContentValues, который предоставляется стандартной android-библиотекой, заносится ключ с названием поля бд, которое необходимо обновить и передаваемое значение – переменная stars. После обновления базы данных, метод updateTimeDBHelpers обновляет список.
Также метод принимает String catId – id персонажа, полученного за прохождение уровня. Данный параметр помечен аннотацией @Nullable т.к. не за все уровни выдается награда.

```
private static void updateTimeStars(double time, int maxThree, int maxTwo, int stars, int whereClause, @Nullable String catId, Mai
    ContentValues cv = new ContentValues();
    if (time < maxThree) stars = 3;
    if (time > maxThree && time < maxTwo) stars = 2;
    if (time > maxTwo) stars = 1;

    cv.put("stars", stars);

    if (timeLevels.get(whereClause - 1).getStars() < stars)
        timeDB.update( table: "time", cv,  whereClause: "_id = " + whereClause,

    updateTimeDBHelpers();
```

IT ШКОЛА SAMSUNG

Класс BasicGameSupport. Статические функции и переменные для работы игры

Класс BasicGameSupport является хранилищем для статических функций и переменных, которые поддерживают работу игры. Так, здесь определены методы для обновления баз данных, перемещения объектов относительно игрока в уровнях на время, нахождения анимации персонажей по ключу и множество других. значений.

Здесь также определены переменные спрайтовых анимаций, постоянных значений.

```
public static int updateMovesY(GameItem gameItem, double jumSpeed, int y) {
public static int updateMovesX(double speedPlayer, int x) {
public static boolean movingRight(MainRunActivity mainRunActivity) {
public static boolean movingLeft(MainRunActivity mainRunActivity) {
private static void updateTimeStars(double time, int maxThree, int maxTwo, int stars, in
public static void updateStrategyMathsStars(int lives, int requestedLives, in
```

```
    ContentValues cv = new ContentValues();
    cv.put("unlocked", 1);
    catsDB.update( table: "cats", cv,  whereClause: "_id = " + whereClause,  whereArgs: null);
    updateCatDBHelpers();
}

//Добавляет кота в комнату (обновляет поле room в бд cats)
public static void putCatIntoRoom(int room, int whereClause) {
    ContentValues cv = new ContentValues();
    cv.put("room", room);
    catsDB.update( table: "cats", cv,  whereClause: "_id = " + whereClause,  whereArgs: null);
    updateCatDBHelpers();
}

//Находит набор анимаций персонажа по полученному ключу
public static ImageSet checkKey(String key) {
    for (ImageSet im : imageSets) if (key.equals(im.getKey())) return im;
    return null;
```

IT ШКОЛА SAMSUNG

Меню, выбор уровней, победа и поражение

MenuView – игровое меню. В нём определены кнопки для перехода в выбор

```
public class ChooseView extends GameView {
    public static PlayerManager playerManager;
    private LevelChoice levelChoice;
    private BasicButton goBack;
    private String key;
    private Bitmap background;

    public ChooseView(MainRunActivity mainRunActivity, String key) {
        super(mainRunActivity);
        this.key = key;

        Bitmap button = null, buttonClicked = null, goBackButton = null, goBackButtonClicked = null;
        background = null;
```

Код, представленный выше – меню с выбором уровней, тип уровней определяется по переданному ключу. В нём представлены 2 переменная **LevelChoice** , которая основываясь на переданном ключе, произведет отрисовку кнопок для выбора уровня, получит данные из базы и установит на каждой кнопке количество звезд.

```
}
levelChoice = new LevelChoice( gameView: this, mainRunActivity, button, buttonClicked, key);
goBack = new BasicButton(mainRunActivity,  x: 20,  y: 540, "Back", Color.BLACK,  size: 30, goBa
```

LevelChoice содержит списки из доступных уровней каждого типа в соответствии с полученным ключом выбирает к какому списку необходимо обратиться и задает соответствующий дизайн. Уровень доступен, если предыдущий пройден т.е. если у (i-1) уровня есть хотя бы 1 звезда.

```
public class LevelChoice implements Loopable {
    private ArrayList<LevelButton> levelButtons;
    private ArrayList<GameView> timeLevels, strategyLevels, mathsLevels;
    private GameView gameView;
    private String key;
```

Уровни на время, стратегические и математические
(все уровни наследуют GameView),

IT ШКОЛА SAMSUNG

Победа и поражение

GameOverView запускается при поражении (если кончились жизни или истекло время). Ниже представлен конструктор данного класса. Он принимает активность, текущий уровень и ключ. Основываясь на ключе, класс определяет к какому типу относится уровень, а с помощью полученного уровня GameView level позволяет повторить попытку прохождения.

```
public GameOverView(MainRun mainRun, GameView level, String key) {
    super(mainRun);
    this.level = level;
    this.key = key;
    tryAgaing = new BasicButton(mainRun,  x: 300,  y: 200, "Play again", Col
    goBack = new BasicButton(mainRun,  x: 300,  y: 400, "Back", Color.BLACK,
```

```
public void repaint() {
    if (tryAgaing.isClicked()) {
        super.getMainRunActivity().setView(level);
        tryAgaing.notClicked();
    }
    if (goBack.isClicked()) {
        super.getMainRunActivity().setView(new ChooseView(super.getMainRunActivity(), key));
        goBack.notClicked();
    }
```

Повторить попытку

Выйти в меню

CongratsView отображается после прохождения уровня. Он так же, как и **GameOverView**, определяет тип уровней по ключу, номер уровня, количество звёзд и награду, представленную объектом Cat.

CongratsView используется только внутри методов, обновляющих звёзды. То есть, метод, обновляющий базу данных, сразу после выполнения основного кода задает новый игровой кадр. Ниже представлен отрывок кода одного из методов, вызывающего CongratsView

```
public static void updateStrategyMathsStars(int lives, int requestedLives, int stars, int whereClause, @Nullable String catId, MainRunActivi
    ContentValues cv = new ContentValues();
    if (lives == requestedLives) stars = 3;
    if (lives < requestedLives && lives > 1) stars = 2;
    if (lives == 1) stars = 1;
    cv.put("stars", stars);

    switch (key) {
        case "strategy":
            if (strategyLevels.get(whereClause - 1).getStars() < stars)
                strategyDB.update( table: "strategy", cv,  whereClause: "_id = " + whereClause,  whereArgs: null);
            updateStrategyDBHelpers();
            if ((catId != null) && (listOfCats.get(Integer.parseInt(catId) - 1).getUnlocked() != 1))
                mainRunActivity.setView(new CongratsView(mainRunActivity, strategyLevels.get(whereClause - 1).getNumber(), stars, listOfCats
            else
                mainRunActivity.setView(new CongratsView(mainRunActivity, strategyLevels.get(whereClause - 1).getNumber(), stars,  reward: nul
            break;
```

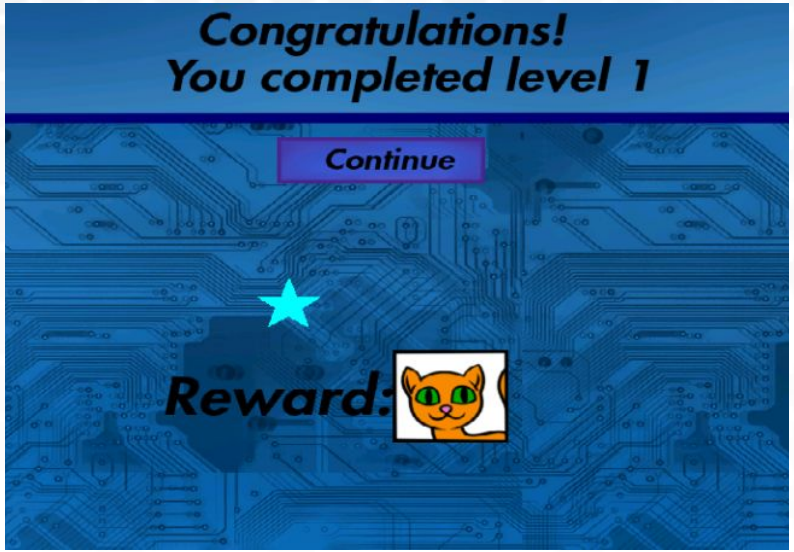
IT ШКОЛА SAMSUNG

Комнаты

```
public CongratsView(MainRun mainRun, int level, int stars, @Nullable Cat reward, String key) {
    super(mainRun);
    this.level = level;
    this.stars = stars;
    this.reward = reward;
    this.key = key;
    returnBack = new BasicButton(mainRun,  x: 300,  y: 150, "Continue", Color.BLACK,  size: 30, baseButton, baseButtonClicked,  stepX: 40,  stepY: 35);
```

Конструктор GongratsView

GameOverView



CongratsView

IT ШКОЛА SAMSUNG

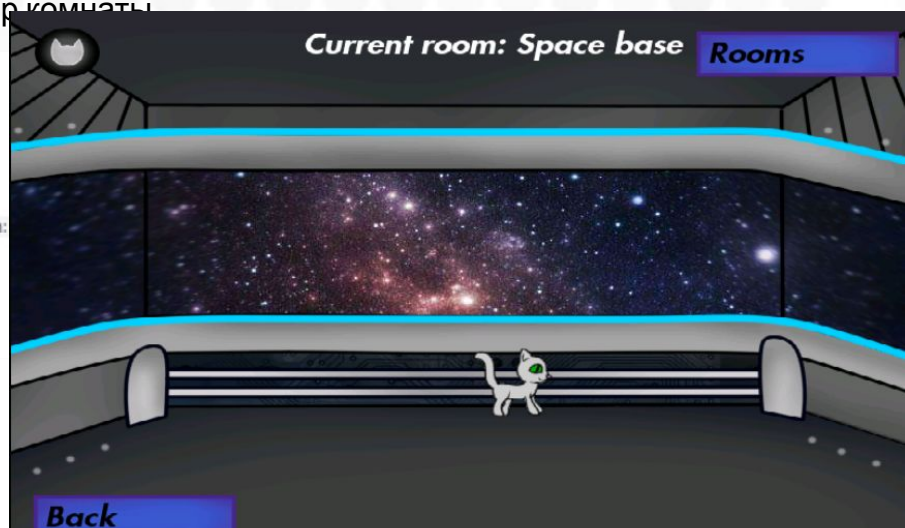
Выбор и размещение персонажей

Чтобы играть за котов, полученных за прохождение уровней, необходимо разместить их в комнату. Комнату отображает RoomView, а логика её работы реализована в вспомогательном классе CurrentRoom.

```
RoomView(MainRun mainRun) {
    super(mainRun);
    rooms = new ArrayList<>();
    roomsChoice = new ArrayList<>();
    goBack = new BasicButton(mainRun,  x: 20,  y: 540, "Back", Color.BLACK,  size: 30, baseButton, baseButtonClicked,
    changeRoom = new BasicButton(mainRun,  x: 610,  y: 25, "Rooms", Color.BLACK,  size: 30, baseButton, baseButtonCli
    rooms.add(new CurrentRoom(mainRun,  number: 1, mainRun.getString(R.string.space base), roomBackground));
```

CurrentRoom – текущая комната со своими параметрами. В комнате 3 свободных места, которые представлены объектом **CatPet**, который в качестве параметров принимает Cat, его id и номер комнаты.

```
public CurrentRoom(MainRun mainRun, int number, String name, Bitmap view) {
    this.mainRun = mainRun;
    this.number = number;
    this.name = name;
    this.view = view;
    places = 0;
    add = new BasicButton(mainRun,  x: 20,  y: 20, addCatButton, addCatButtonClicked,  item:
    catPet1 = new CatPet( cat: null,  id: -1, number);
    catPet2 = new CatPet( cat: null,  id: -1, number);
    catPet3 = new CatPet( cat: null,  id: -1, number);
    catStorage = new ArrayList<>();
```



IT ШКОЛА SAMSUNG

Выбор и размещение персонажей

```
for (int i = 0; i < catStorage.size(); i++) {
    if (catStorage.get(i).isClicked() && listOfCats.get(i).getUnlocked() == 1 &

        BasicButton add = new BasicButton(mainRunActivity, x: 115, y: catStorag
        if (listOfCats.get(i).getRoom() == -1) add.run(gamePaint);

        BasicButton choose = new BasicButton(mainRunActivity, x: 115, y: catSto
        choose.run(gamePaint);

        if (add.isClicked() && places < 3) {
            //Добавить кота в данную комнату
            BasicGameSupport.putCatIntoRoom(number, listOfPets.get(i).getId());
            if (listOfCats.get(i).getUnlocked() == 1) checkCatPet(listOfPets.ge
            catStorage.get(i).notClicked();
            add.notClicked();
        } else add.notClicked();

        if (choose.isClicked() && listOfCats.get(i).getRoom() != -1) {
            BasicGameSupport.choosePlayer(listOfCats.get(i).getId());
            PlayerManager.setChosenCat(listOfCats.get(i));
            catStorage.get(i).notClicked();
            choose.notClicked();
        } else choose.notClicked();
    } else catStorage.get(i).notClicked();
}
```

Кота можно добавить в комнату, если он доступен(поле unlocked в бд cats = 1) и в комнате есть свободные места. Тогда , при нажатии на иконку персонажа , всплывают 2 кнопки – добавить в комнату и выбрать. При добавлении, поле room заменяется на номер комнаты, обновление базы производит статический метод putCatIntoRoom. Статический метод setChosenCat заменяет текущего выбранного персонажа на нового. TimePlayer(класс игрока в уровнях на время) берет характеристики и анимации того кота, который помечен как chosen = 1 в бд cats.

IT ШКОЛА SAMSUNG

Уровни на время

Все уровни на время различны – у них разные препятствия, требования для прохождения, сложность, длительность. Но всё же они во многом схожи. Для описания всех базовых параметров уровней на время, а также для объединения их в единую сущность создан абстрактный класс TimeLevel. Конструктор класса:

```
protected TimeLevel(int twoStars, int threeStars, double totalTime, Bitmap background, Bitmap ground, int lives) {
    this.twoStars = twoStars;
    this.threeStars = threeStars;
    this.totalTime = totalTime;
    this.lives = lives;
    movingBackground = new MovingBackground(background,  speed: 3);
    timeGround = new TimeGround(ground);
    nowTime = System.nanoTime() / BasicGameSupport.SECOND;
}
```

Конструктор принимает требуемое время для получения 2 и 3 звезд, изображения фона и земли в уроне, кол-во жизней.

Обращение одного из наследников к конструктору TimeLevel

```
public Level1(MainRunActivity mainRunActivity) {
    super( twoStars: 20,  threeStars: 25,  totalTime: 30, movingSpaceBackground, blueGround,  lives: 1);
```

TimeLevel также предоставляет методы для начальной и конечной отрисовки уровня, repaint для обновления базовых параметров и несколько других методов для поддержки мини-игры «ракета».

```
@Override
public void run(GamePaint gamePaint) {
    //Начальная отрисовка (под всеми игровыми элементами)
    movingBackground.run(gamePaint);
    timeGround.run(gamePaint);
    passingDoor.repaint();
}

protected void endingRun(GamePaint gamePaint) {
    //Конечная отрисовка (над всеми игровыми элементами)
    if (lives <= 0) gameOver = true;
    endTime = System.nanoTime() / BasicGameSupport.SECOND -
```

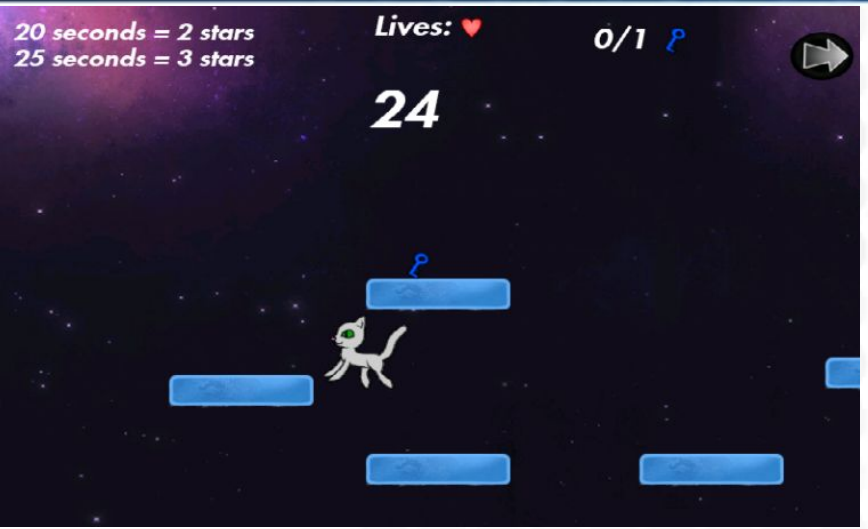
Класс содержит всего один абстрактный метод – BasicGameSupport.isRequirementsCollected

```
public abstract boolean isRequirementsCollected();
```

Во время прохождения уровней, метод timeLevelFinish проверяет, пройден ли уровень, а для этого важно знать требования к прохождению. Так, во 2 уровне требований нет, а в 5 уроне необходимо собрать 20 ключей. Поэтому метод является абстрактным и определенном именно в суперклассе : каждый класс реализовывает его по-разному, а благодаря принадлежности метода к общему родительскому классу, метод для проверки прохождения уровня становится универсальным.

IT ШКОЛА SAMSUNG

Уровни на время

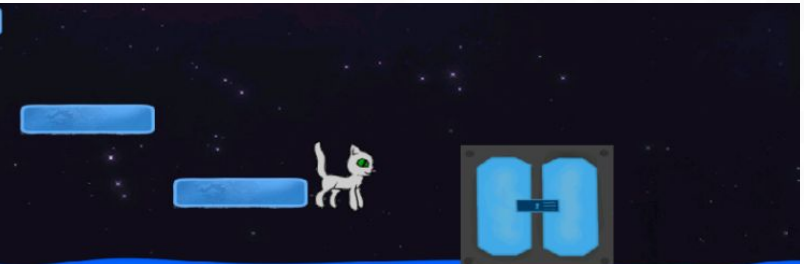


В уровнях на время нужно успеть добежать до финиша за определенное время. Логика уровней определяется в классах Level1, Level2… и т.д. , которые наследуют класс TimeLevel. Для поддержки уровней, все объекты определены в директории gameTime. В ней находятся классы, описывающие игрока, препятствия, прочие предметы.

Игрок прописан в классе TimePlayer. Предметы, препятствия, с которыми взаимодействует игрок используют представленные ниже методы для обновления координат по оси x (при движении игрока горизонтально) и по оси y (при движении вертикально, после достижения определенного расстояния).

```
ublic static int updateMovesX(double speedPlayer, int x) {
    if (!timePlayer.isCollisionDetectedRight()) if (timePlayer.isMovingLeft()) x += speedP
    if (!timePlayer.isCollisionDetectedLeft()) if (timePlayer.isMovingRight()) x -= speedP

    return x;
public static int updateMovesY(GameItem gameItem, double jumSpeed, int y) {
    if (timePlayer.isJumpingLimit()) {
        if (timePlayer.isJumping()) y += jumSpeed;
        if (timePlayer.isFalling() && !timePlayer.isJumping()) y -= jumSpeed;
    } else {
        if (y < gameItem.getControlY()) y += jumSpeed;
        if (y > gameItem.getControlY()) y -= jumSpeed;
    }
    return y;
```

Для определения столкновений используются коллизии, о которых было сказано раннее.



В уровне 1, который показан на скриншоте, необходимо взять ключ и открыть дверь. 20 потраченных секунд = 3 звезды, 25 = 2 звезды, 1 звезда, если просто удалось уложиться во время.

IT ШКОЛА SAMSUNG

Мини-игра «Ракета»

Мини-игра «Ракета» является дополнением уровней на время. Запрыгнув на специальную платформу, кот взлетает на ракете и задачей игрока становится увернуться от препятствий и собрать необходимые предметы.

У класса TimePlayer есть булевая переменная rocketMode. Если rocketMode = true, то запускается специальный метод для обновления и отрисовки игрока.

В игре есть 3 дороги, по которым нужно перемещаться при помощи свайпов вверх и вниз. Расчет свайпов производится в

```
if (mainRunActivity.getTouchListener().isSwipeUp() && y - 130 > 200) {
    y -= 120;
    mainRunActivity.getTouchListener().setSwipeUp(false);
}
if (mainRunActivity.getTouchListener().isSwipeDown() && y + 120 <= 600) {
    y += 120;
    mainRunActivity.getTouchListener().setSwipeDown(false);
}
```

Данный отрывок кода показывает реализацию перемещения по дорожками с помощью свайпов

Метод generateRocketItems, определенный в классе TimeLevel генерирует предметы и препядствия в уровне , заносит их в ArrayList'ы,

производит удаление объектов, если они вылетели за пределы экрана.

```
generateRocketItems(ArrayList<TimeInventoryItem> itemsG, ArrayList<TimeInven
                    @NonNull ArrayList<TimeInventoryItem> badItems, int[] re
```

Добавление новых объектов в соответствующие списки:

```
if (itemsB.size() >= 1 && itemsB.size() < 20 && itemsB.get(itemsB.size() - 1).getX() <= 750)
    itemsB.add(bad);
else if (itemsB.size() < 1)
    itemsB.add(bad);

if (itemsG.size() >= 1 && itemsG.size() < 3 && itemsB.get(itemsB.size() - 1).getX() == 850)
    itemsG.add(good);
else if (itemsG.size() < 1)
    itemsG.add(good);
```

IT ШКОЛА SAMSUNG

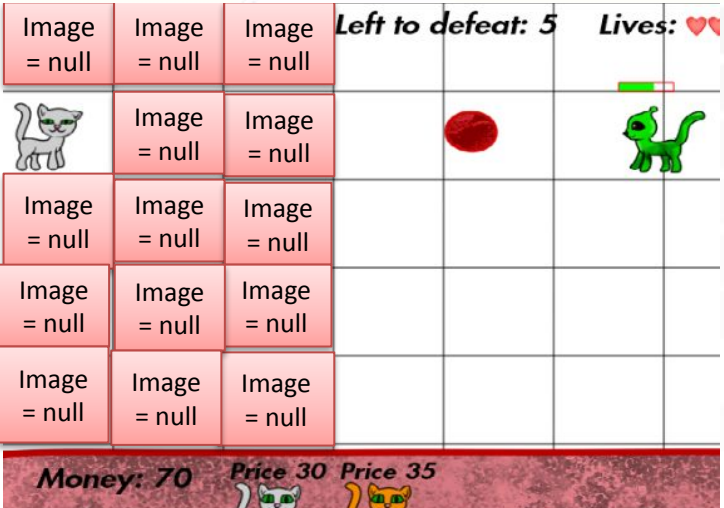
Стратегические уровни

gameStrategy
- StrategyBullet
- StrategyEnemy
- StrategyField
- StrategyLevel
- StrategyPlayer

Все стратегические уровни, в отличие от временных, где каждый уровень – новое игровое поле со своими индивидуальными функциями, имеют схожую логику. Нет смысла в создании новых классов ради того, чтобы изменять силу врагов или количество денег игрока. Поэтому все непостоянные значения передаются в качестве параметров классу **StrategyField**, в котором определена логика игрового поля. Перед тем, как StrategyField начнём выполнение основного кода, запускается класс **CatChoice**, который просит выбрать от 1 до 5 персонажей для игры.

Изначально StrategyPlayer-ы инициализируются cmd ImageSet, всеми параметрами конкретного кота(сила, id, задержка) -1. StrategyPlayer реагирует на касания как кнопки - clicked true/false, срабатывать обработчик начинает только тогда, когда imageSet!=null т.е. после добавления кота в экземпляр StrategyPlayer. Внутри StrategyPlayer'a сразу определен список экземпляров класса **StrategyBullet** – пули, которыми стреляет кот с определенной задержкой.

```
players.add(new StrategyPlayer( id: -1, mainRun,  image: null,  power: -1,  delay: -1, x, y,  priceDelay: 5,  pricePower: 7, bullet, count));
```




```
private ArrayList<BasicButton> characters;
private ArrayList<StrategyPlayer> players;
```

Часть игрового поля 3*5– список объектов **StrategyPlayer**.

IT ШКОЛА SAMSUNG

Стратегические уровни

Добавление кота в клетку: Сначала проверяем нажата ли i-я кнопка, если да – ждём пока игрок выберет клетку, в которой нужно разместить персонажа. В выбранную клетку заносятся все необходимые данные кота – его id, изображение, сила, скорость. Эта информация получается из объекта класса CatChoice, который, в свою очередь, получает данные из бд с персонажами.

```
playerButtons.get(i).isClicked())
if (money >= catChoice.getChosenStrategyCats().get(i).getPrice()) {
    for (int j = 0; j < players.size(); j++)
        if (players.get(j).isClicked()) {
            if (players.get(j).getImage() == null) {
                players.get(j).setId(catChoice.getChosenStrategyCats().get(i).getId());
                players.get(j).setImage(players.get(j).getImageSets().get(catChoice.getChosenStrat
                players.get(j).setPower(catChoice.getChosenStrategyCats().get(i).getPower());
                players.get(j).setDelay(catChoice.getChosenStrategyCats().get(i).getDelay());
                money -= catChoice.getChosenStrategyCats().get(i).getPrice();
                players.get(j).notClicked();
                playerButtons.get(i).notClicked();
            } else {
```

Получение характеристик i-го персонажа из CatChoice

Стрельба игрока с задержкой delay

```
if (image != null) {
    image.getStandRight().run(gamePaint, x, y, delay: 2);

    if (easyTimer.timerDelay(delay)) {
        strategyBullets.add(new StrategyBullet(x, y, delay, bullet))
        easyTimer.startTimer();
    }

    for (int i = 0; i < strategyBullets.size(); i++) {
        strategyBullets.get(i).run(gamePaint);
        if (strategyBullets.get(i).getX() > 1000) strategyBullets.re
    }
```

Конструктор StrategyField

```
public StrategyField(MainRunActivity mainRunActivity, int enemiesCount,
                     int money, double lowerDelay, double powerIncrease,
                     int leftToDefeat, int lives, int enemyReward, int[] enemyIds,
                     int speed, @Nullable String rewardId) {
```

IT ШКОЛА SAMSUNG

Математические уровни

gameMathematics
- MathsAnswer
- MathsField
- MathsPlayer
- Theory

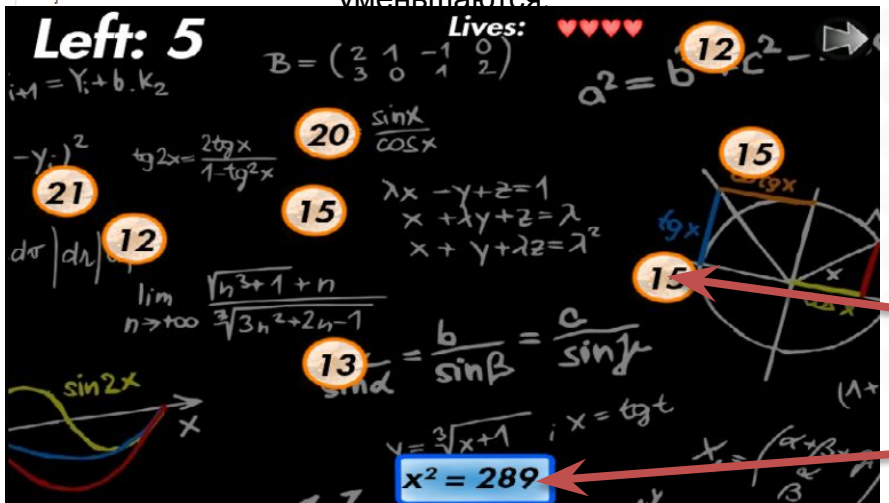
Принцип работы математических уровней аналогичен стратегическим. Основная логика игры прописана в классе **MathsField**. В классе **Theory** хранится информация с математической теорией для игры. MathsPlayer – «игрок» , который будет ловить ответы. **MathsAnswer** – ответы.

```
for (int i = 0; i < mathsAnswers.size(); i++) {
    if (checkTwoItemCollision(mathsPlayer, mathsAnswers.get(i))) {
        if (theory.checkMathAnswer(mathsPlayer.getExpression(), mathsAnswers.get(i).getExpression())) {
            setQuestion();
            total--;
        } else {
            lives--;
        }
        mathsAnswers.remove(i);
    }
```

В MathsField сразу после столкновения ответа и игрока производится проверка:
Если ответ подходит , количество совпадений для победы уменьшается, если же нет – жизни уменьшаются.

```
boolean checkMathAnswer(String expr, String answer) {
    int indexQ = expr.indexOf("?");
    String result = "";
    if (indexQ == -1)
        result = expr.replace( target: "x", answer);
    else
        result = expr.replace( target: "?", answer);

    for (int i = 0; i < expressions.size(); i++) {
        if (result.equals(expressions.get(i))) {
            return true;
```

Проверка столкновения игрока и ответа. Метод checkMathsAnswer заменяет «x» или «?» в вопросе на полученные ответ. Если полученное выражение совпадает с одним из теории, метод вернет true, в противном случае – false.

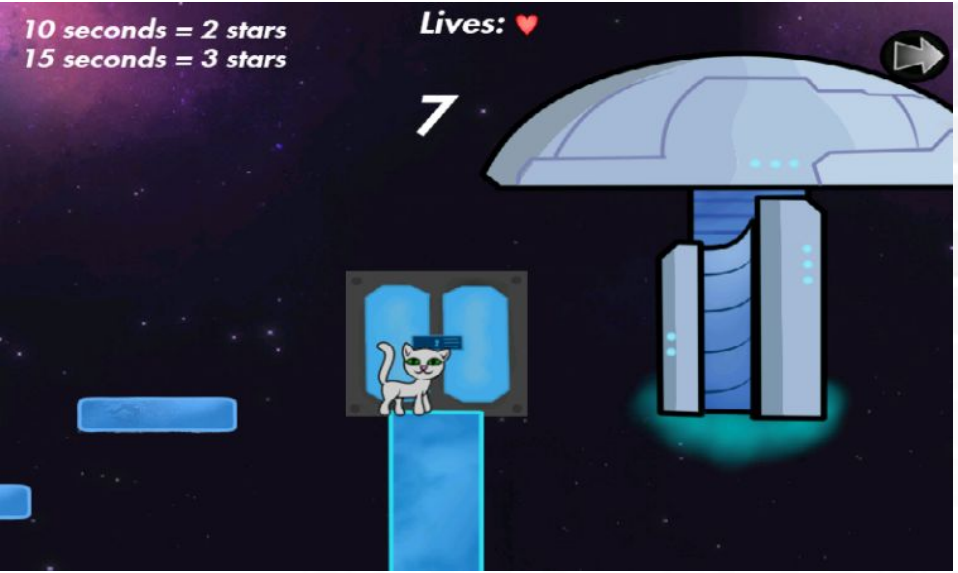


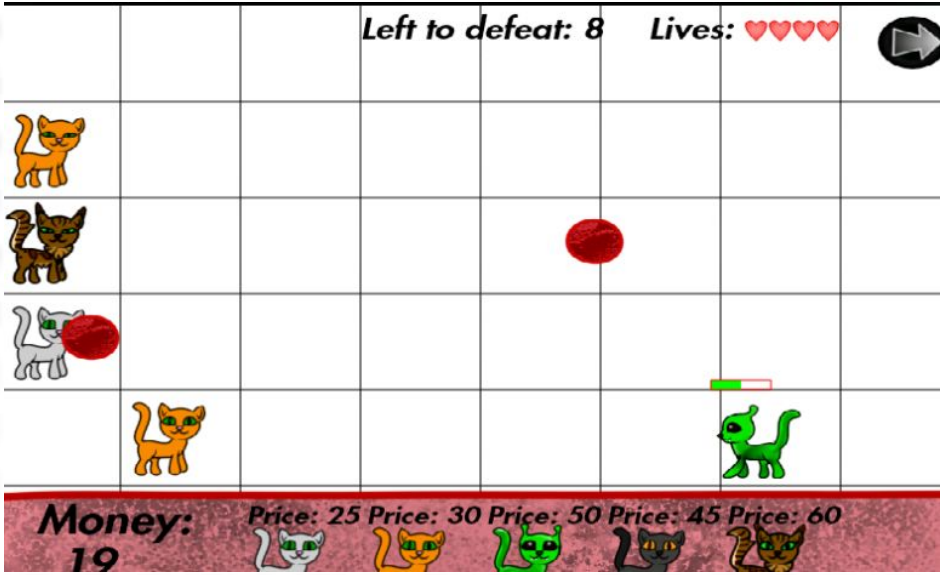
MathsAnswer

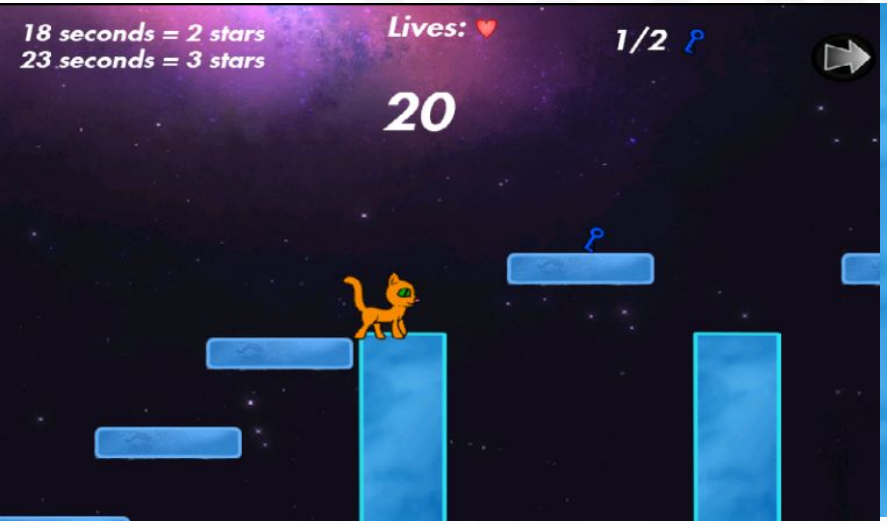
MathsPlayer

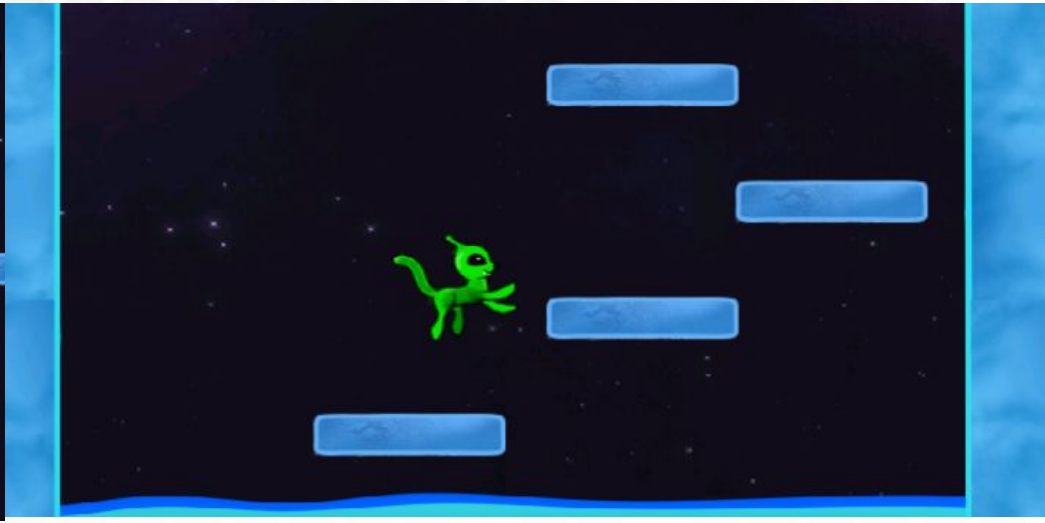
IT ШКОЛА SAMSUNG

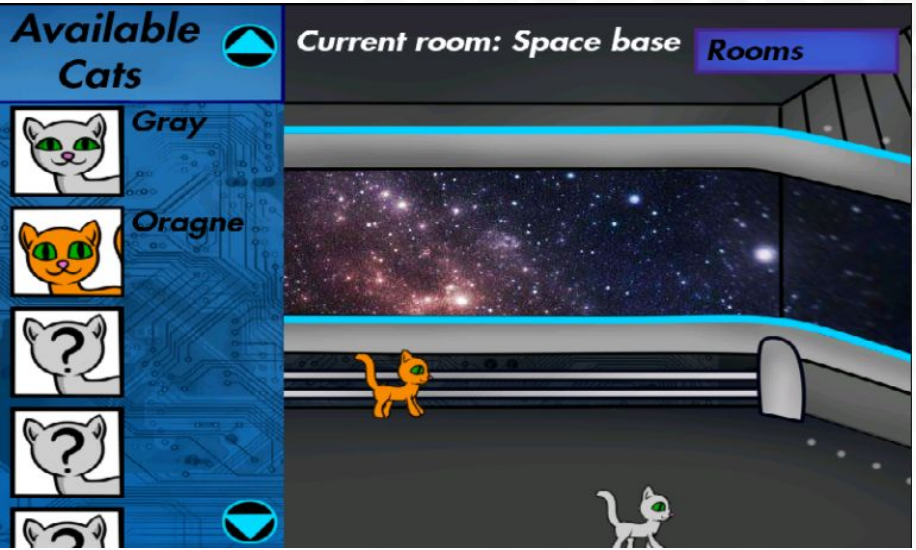
Кадры из игры

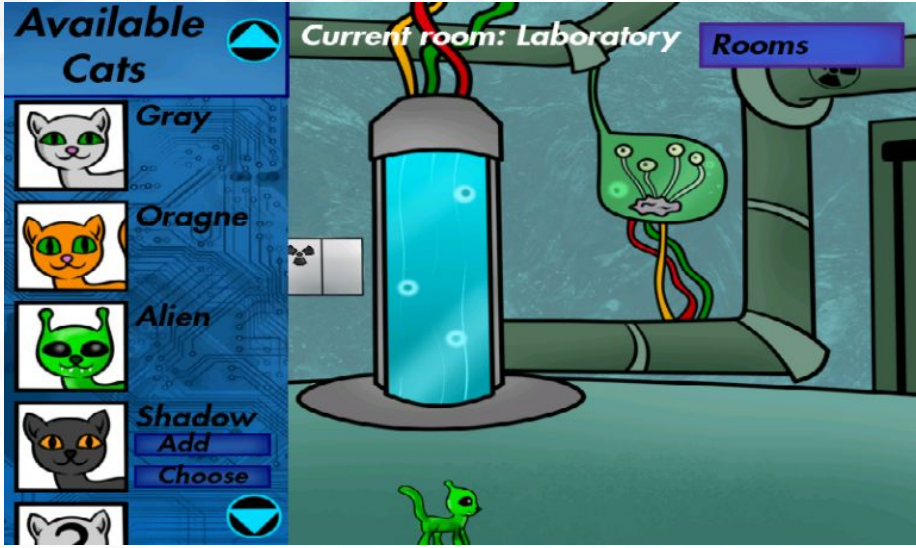


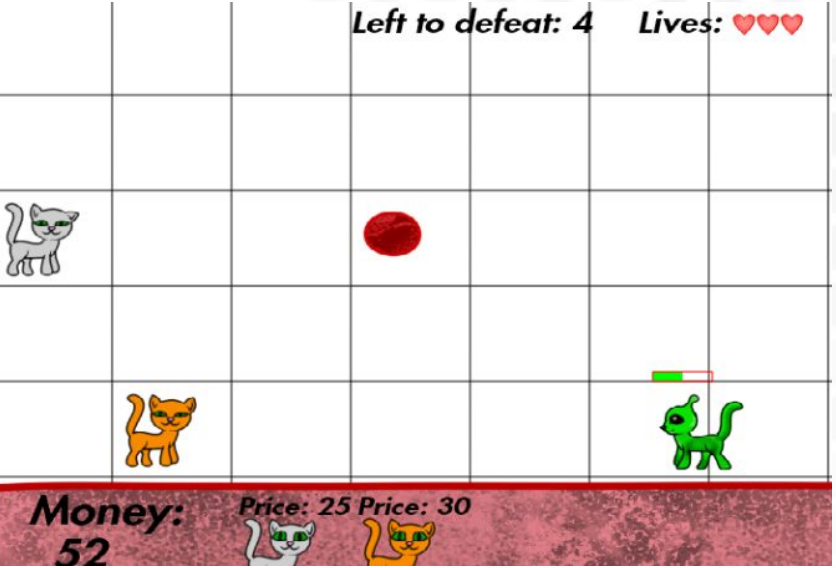


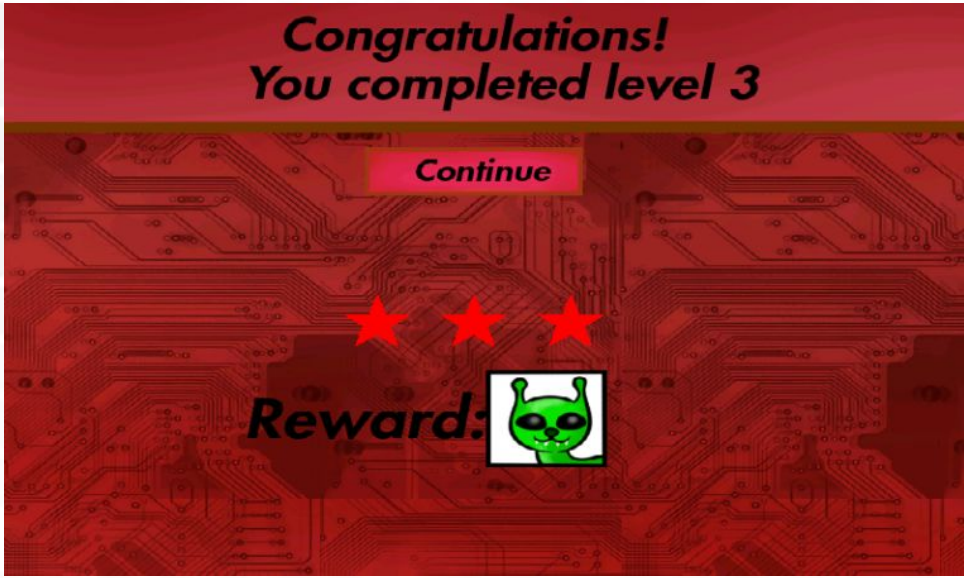

18 seconds = 2 stars
23 seconds = 3 stars
Lives:
1/2
20

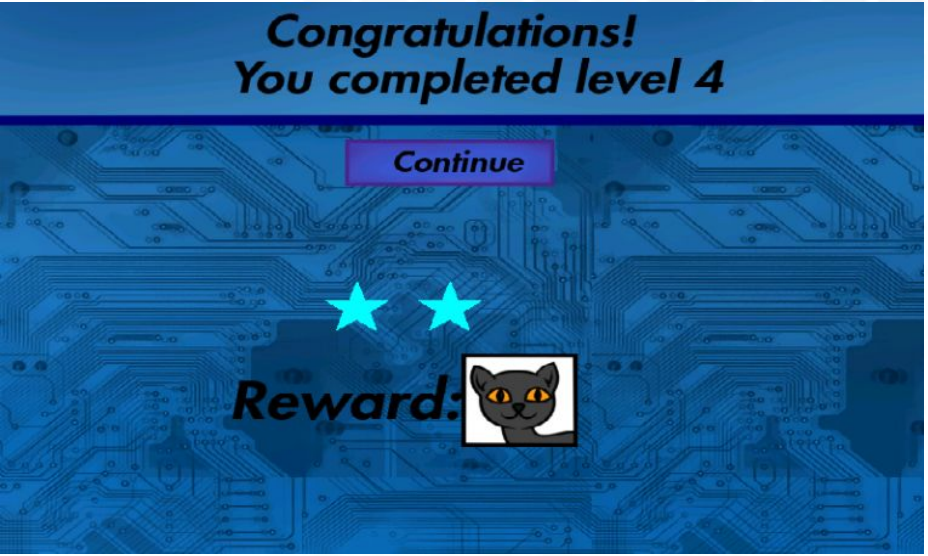


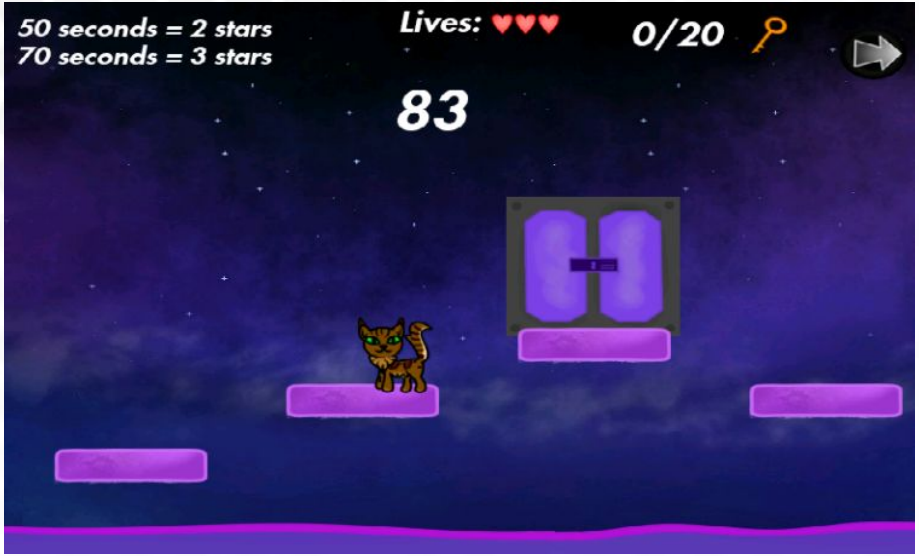

Available Cats
Gray
Oragne
Current room: Space base
Rooms

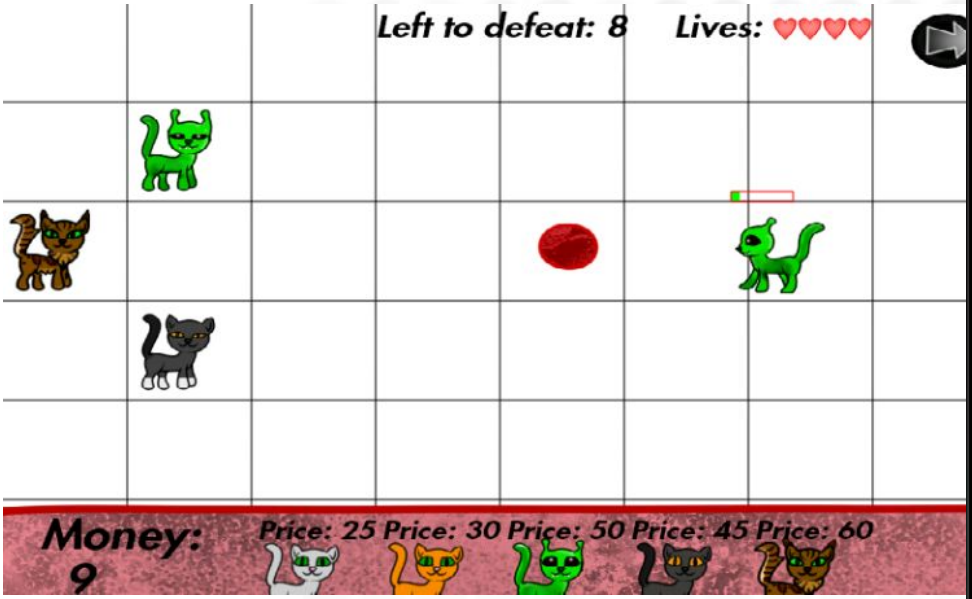

Available Cats
Gray
Oragne
Alien
Shadow
Add
Choose
Current room: Laboratory
Rooms

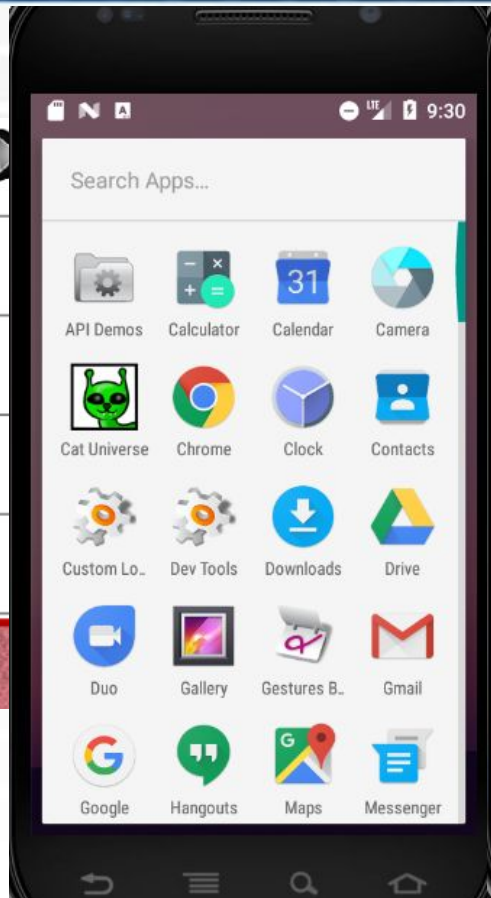

Left to defeat: 4
Lives:
Money:
52
Price: 25
Price: 30

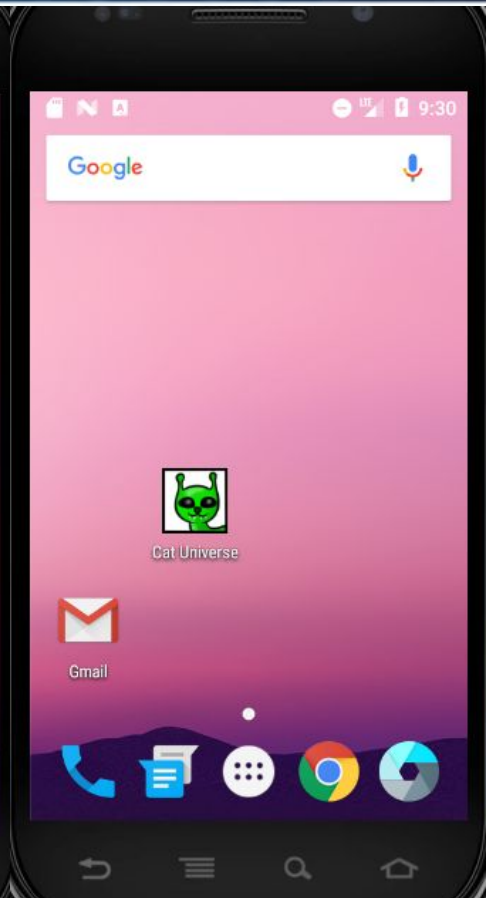

Congratulations!
You completed level 3
Continue
Reward:


Congratulations!
You completed level 4
Continue
Reward:


50 seconds = 2 stars
70 seconds = 3 stars
Lives:
0/20
83

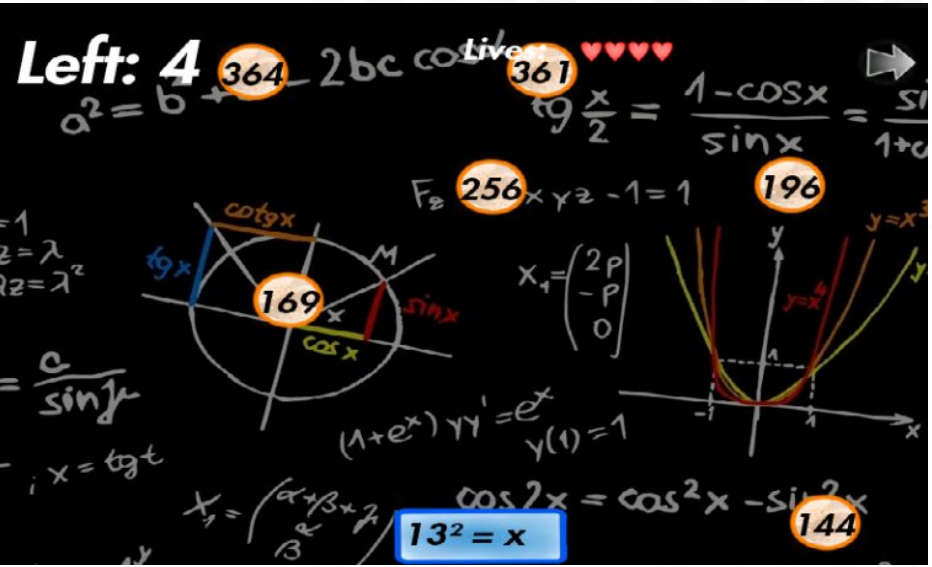

Left to defeat: 8
Lives:
Money:
9
Price: 25 Price: 30 Price: 50 Price: 45 Price: 60

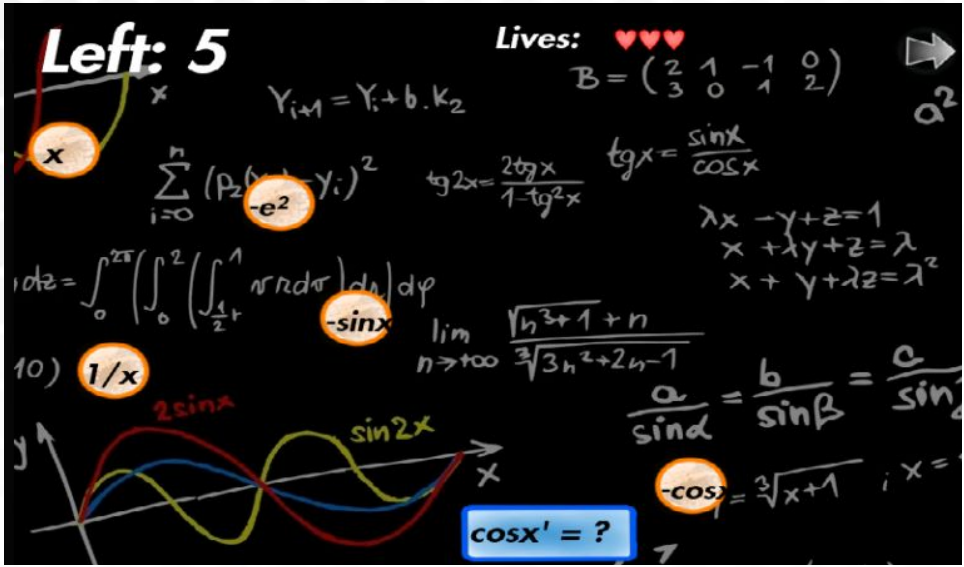

9:30
Search Apps...
API Demos
Calculator
Calendar
Camera
Cat Universe
Chrome
Clock
Contacts
Custom Lo..
Dev Tools
Downloads
Drive
Duo
Gallery
Gestures B..
Gmail
Google
Hangouts
Maps
Messenger


9:30
Google
Cat Universe
Gmail

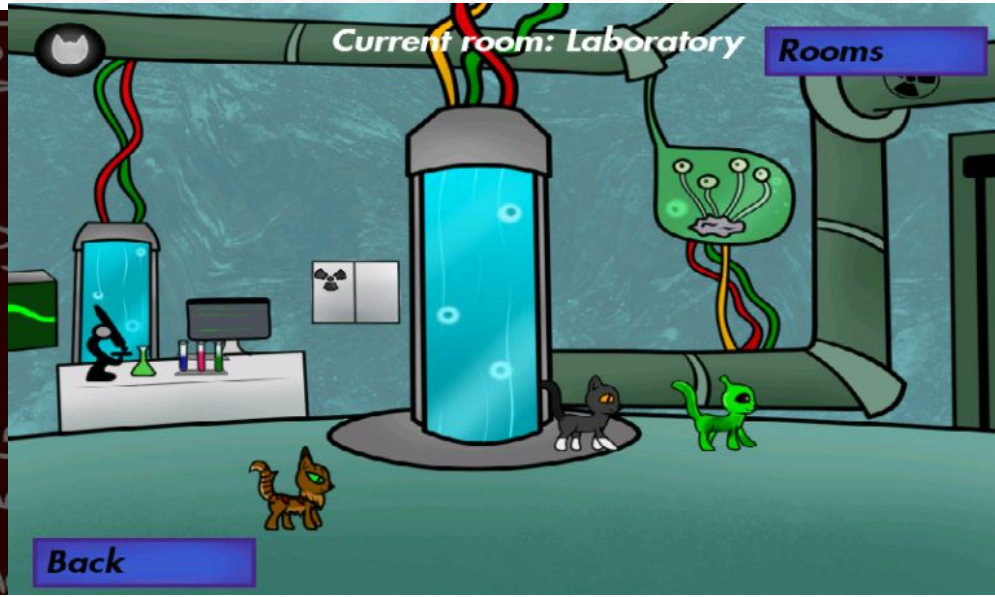

50 seconds = 2 stars
70 seconds = 3 stars
Lives:
0/20
64


50 seconds = 2 stars
70 seconds = 3 stars
Lives:
2/20
4


Left: 4
Lives:
364
361
256
196
169
144
$13^2 = x$


Left: 5
Lives:
x
$-e^2$
-sinx
1/x
2sinx
sin2x
$\cos x' = ?$


Congratulations!
You completed level 3
Continue
Reward:


Current room: Laboratory
Rooms
Back

Итоги

Дальнейшее планы развитие приложения:

Улучшение графического интерфейса путём увеличения размеров главного bitmap'a.
Добиться максимальной гибкости кода, сократить его.

Создание интересных возможностей для уровней : суперпрыжки, двигающиеся препятствия в уровнях на время, новые враги, другие типы усиления в стратегических уровнях , различные усилители в математических уровнях. Это доработки сделают игру интереснее для пользователей.

Выставить игру в PlayMarket.

Заключение:

На создание проекта немало времени, пришлось справиться с десятками ошибок, не раз приходилось стирать весь код в некоторых классах и писать с чистого листа. Но я не жалею о потраченном времени - это бесценный опыт для меня, ведь теперь я могу писать то, на создание чего у меня уходили часы, в разы быстрее, идеи, которые приходили ко мне во время создания игры, могут использоваться в дальнейшем, в моих новых проектах.

Само приложение при должных доработках способно собрать аудиторию из любителей аркадных и стратегических игр, помочь школьникам освоить теорию математики , в игровом виде. За математические уровни будут давать интересных персонажей, поэтому если школьник захочет собрать всех, ему придётся погрузиться в мир математики. Возможно, после этого, успеваемость школьника повысится и он поймёт, что математика вовсе не скучная, а интересная и важная наука.

IT ШКОЛА SAMSUNG

Благодарность

Хочу сказать огромное спасибо за проведение такого замечательного курса! Мне даже жаль, что он подошел к концу. Во время занятий мне удалось изучить много нового, а также закрепить те темы, с которыми я уже была знакома раннее. Отдельное спасибо нашему преподавателю, Рудину Павлу, за отличное преподнесение информации, заинтересованность, желание помочь, подсказать каждому ученику курса!

Желаю, чтобы в будущем в it школу samsung приходило всё больше талантливых, увлечённых учеников, желающих учиться и развиваться в it сфере!

СПАСИБО ЗА ВНИМАНИЕ