**Исходный текст программы**

**Программный модуль GameState.java**

package game.me.socoban.gamecontrol;

import java.io.Serializable;

import java.util.LinkedList;

import game.me.socoban.constants.Levels;

/\*

\* класс для работы с состоянием игры

\* текущее положение

\* проверка на сходимость

\* проверка на перемещение и т.п.

\* \*/

public class GameState implements Serializable {

static class Undo implements Serializable { // класс для сериализации истории ходов

public char c1; // начальное положение

public char c2; // конечно положение

public char c3; // смещение коробки

public byte x1;

public byte x2;

public byte x3;

public byte y1;

public byte y2;

public byte y3;

}

public static final char CHAR\_BOX\_ON\_FLOOR = '$'; //ящик на игровом поле

public static final char CHAR\_BOX\_ON\_ENDPOINT = '\*'; //ящик на месте размещения (цели)

public static final char CHAR\_FLOOR = ' '; // игровое поле

public static final char CHAR\_PLAYER\_ON\_FLOOR = '@'; //пользователь стоит на поле

public static final char CHAR\_PLAYER\_ON\_TARGET = '+';//пользователь стоит на месте размещения

public static final char CHAR\_TARGET = '.';//цель размещения

public static final char CHAR\_WALL = '#';//ограничевающия стена

// проверяем куда сдвинули ящик на пусто или на цель

private static char newCharWhenBoxPushed(char current) {

return (current == CHAR\_FLOOR) ? CHAR\_BOX\_ON\_FLOOR : CHAR\_BOX\_ON\_ENDPOINT;

}

private static char newCharWhenPlayerEnters(char current) {

switch (current) {

case CHAR\_FLOOR:

case CHAR\_BOX\_ON\_FLOOR:

return CHAR\_PLAYER\_ON\_FLOOR;

case CHAR\_TARGET:

case CHAR\_BOX\_ON\_ENDPOINT:

return CHAR\_PLAYER\_ON\_TARGET;

}

throw new RuntimeException("Неверный символ: '" + current + "'");

}

//проверка когда игрок сместился - под ним было цель или пусто

private static char originalCharWhenPlayerLeaves(char current) {

return (current == CHAR\_PLAYER\_ON\_FLOOR) ? CHAR\_FLOOR : CHAR\_TARGET;

}

//массив символов переделываем в матрицу

private static char[][] stringArrayToCharMatrix(String[] s) {

char[][] result = new char[s[0].length()][s.length];

for (int x = 0; x < s[0].length(); x++) {

for (int y = 0; y < s.length; y++) {

result[x][y] = s[y].charAt(x);

}

}

return result;

}

private int currentLevel; //текщий уровень

final int currentLevelSet; //текущий тип игры

private char[][] map; //карта

private transient final int[] playerPosition = new int[2]; //доавбили модификатор исключения из серилизации так как данные не нужно запоминать

final LinkedList<Undo> undos = new LinkedList<Undo>(); //массив хород для отката шага

public GameState(int level, int levelSet) { //класс сотсояния игры и загрузка уровня

currentLevel = level;

currentLevelSet = levelSet;

loadLevel(currentLevel, levelSet);//гразрузка уровня

}

public int getCurrentLevel() {

return currentLevel;

}

public int getHeightInTiles() {

return map[0].length;

}

public char getItemAt(int x, int y) {

return map[x][y];

} //возврат элемента карты по координатам

//позиция игрока

public int[] getPlayerPosition() {

for (int x = 0; x < map.length; x++) {

for (int y = 0; y < map[0].length; y++) {

char c = map[x][y];

if (CHAR\_PLAYER\_ON\_FLOOR == c || CHAR\_PLAYER\_ON\_TARGET == c) {

playerPosition[0] = x;

playerPosition[1] = y;

}

}

}

return playerPosition;

}

//ширина поля

public int getWidthInTiles() {

return map.length;

}

//все коробки на целях

public boolean isDone() {

for (char[] chars : map)

for (int y = 0; y < map[0].length; y++)

if (chars[y] == CHAR\_BOX\_ON\_FLOOR)

return false;

return true;

}

//формируем карту поля

private void loadLevel(int level, int levelSet) {

this.currentLevel = level;

map = stringArrayToCharMatrix(Levels.levelMaps.get(levelSet)[level]);

}

//отмена хода

public boolean performUndo() {

if (undos.isEmpty())

return false;

Undo undo = undos.removeLast(); //из массива сохранения удаляем послее действия

map[undo.x1][undo.y1] = undo.c1; // на координаты вызвращаем положения

map[undo.x2][undo.y2] = undo.c2;

if (undo.c3 != 0)

map[undo.x3][undo.y3] = undo.c3;

return true;

}

//сброс уровня

public void restart() {

loadLevel(currentLevel, currentLevelSet);

undos.clear();

}

// проверка возможности перемещения

public boolean tryMove(int dx, int dy) {

if (dx == 0 && dy == 0)

return false;

if (dx != 0 && dy != 0) {

throw new IllegalArgumentException("Можно двигаться только по прямой линии. dx=" + dx + ", dy=" + dy);

}

int steps = Math.max(Math.abs(dx), Math.abs(dy)); //количнство шагов

int stepX = (dx == 0) ? 0 : (int) Math.signum(dx); //кол-во по х

int stepY = (dy == 0) ? 0 : (int) Math.signum(dy); //кол-во по у

boolean somethingChanged = false; //что то изменилось? флаг

int playerX = -1; //позиция игрока х

int playerY = -1;//позиция игрока у

// ищем положение игрока

for (int x = 0; x < map.length; x++) {

for (int y = 0; y < map[0].length; y++) {

char c = map[x][y];

if (CHAR\_PLAYER\_ON\_FLOOR == c || CHAR\_PLAYER\_ON\_TARGET == c) {

playerX = x;

playerY = y;

}

}

}

//делаем перемещения согласно сдвигу по шагам

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

int newX = playerX + stepX; //новое положения игрока по х

int newY = playerY + stepY;//новое положения игрока по н

boolean ok = false;

boolean pushed = false;

switch (map[newX][newY]) { //проверка куда пришли

case CHAR\_FLOOR:

// движение в пустоту

case CHAR\_TARGET:

// движение к свободной цели

ok = true;

break;

case CHAR\_BOX\_ON\_FLOOR:

// смещение коробки в свободную область

case CHAR\_BOX\_ON\_ENDPOINT:

// смещение коробки к конечной цели

char pushTo = map[newX + stepX][newY + stepY];

ok = (pushTo == CHAR\_FLOOR || pushTo == CHAR\_TARGET);

// если перемещение коробки возможно то

if (ok) {

pushed = true; //флаг сдига коробки

}

break;

}

if (ok) {

Undo undo; //элемент шага к сохранению

if (undos.size() > 2000) {

// устанавливаем размер очереди запоминания действий

// если он превышен то удаляем первый в очереди

undo = undos.removeFirst(); //берем 1 элемент списке

undo.c3 = 0;

} else {

undo = new Undo();

}

undos.add(undo);

somethingChanged = true; // ставим пометку что ход состоялся

if (pushed) { //сдвинули коробку

byte pushedX = (byte) (newX + stepX);

byte pushedY = (byte) (newY + stepY);

undo.x3 = pushedX; //новое положение коробки

undo.y3 = pushedY;

undo.c3 = map[pushedX][pushedY]; // новый символ

map[pushedX][pushedY] = newCharWhenBoxPushed(map[pushedX][pushedY]);

}

undo.x1 = (byte) playerX; // положение игрока до смещения

undo.y1 = (byte) playerY;

undo.c1 = map[playerX][playerY];

map[playerX][playerY] = originalCharWhenPlayerLeaves(map[playerX][playerY]);

undo.x2 = (byte) newX; // положение игрока после смещения

undo.y2 = (byte) newY;

undo.c2 = map[newX][newY];

map[newX][newY] = newCharWhenPlayerEnters(map[newX][newY]);

playerX = newX;

playerY = newY;

if (isDone()) {

// если все коробки на месте - игра пройдена

return true;

}

}

}

return somethingChanged;

}

}

**Программный модуль GameView.java**

package game.me.socoban.gamecontrol;

import android.app.Activity;

import android.app.AlertDialog;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.content.res.Resources;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.graphics.Matrix;

import android.os.Vibrator;

import android.util.AttributeSet;

import android.view.KeyEvent;

import android.view.MotionEvent;

import android.view.View;

import game.me.socoban.GameActivity;

import game.me.socoban.R;

import game.me.socoban.constants.ConstMain;

import game.me.socoban.constants.Levels;

//кастомный view - отображение поля игры

public class GameView extends View {

static class GameMetrics {

boolean levelFitsOnScreen; //уровень уже полностью по всему экрану

int tileSize; // рамзерность изображений для поля

}

final GameState game; //состяние игрового поля

boolean ignoreDrag; //игнорируем смещение

GameMetrics metrics;

private int offsetX; //смещение

private int offsetY;

//рисунки игрока, степ, пустоты, цели, ящика

private Bitmap boxOnFloorBitmap; //риснок ящика на поле

private Bitmap boxOnEndPointBitmap; //рисунок ящика на цели

private Bitmap floorBitmap; // пустое поле (можно наступать)

private Bitmap playerOnFloorBitmap; //игрок на поле

private Bitmap playerOnTargetBitmap; // игрок на цели

private Bitmap outsideBitmap; // пустота за полем

private Bitmap targetBitmap; // цель

private Bitmap wallBitmap; //ограничения поля - стена

public GameView(Context context, AttributeSet attributes) {

super(context, attributes);

if (isInEditMode()) { //проверка что вьюе в режмы изменения

game = null;

return;

}

if (context instanceof GameActivity) { //проверем что контекст действительно принадлежит игровому окну

this.game = ((GameActivity) context).gameState;

} else {

game = null;

return;

}

//описываем дейсвтяи при таче по вью

setOnTouchListener(new OnTouchListener() {

private int xOffset;

private int xTouch;

private int yOffset;

private int yTouch;

@Override

public boolean onTouch(View v, MotionEvent event) {

if (event.getAction() == MotionEvent.ACTION\_DOWN) {

ignoreDrag = false;

xTouch = (int) event.getX();

yTouch = (int) event.getY();

xOffset = 0;

yOffset = 0;

} else if (event.getAction() == MotionEvent.ACTION\_UP) {

} else if (event.getAction() == MotionEvent.ACTION\_MOVE) {

if (ignoreDrag)

return true;

xOffset += xTouch - (int) event.getX();

yOffset += yTouch - (int) event.getY();

int dx = 0, dy = 0;

if (Math.abs(xOffset) >= Math.abs(yOffset)) {

// движение по оси х

dx = (xOffset) / metrics.tileSize;

if (dx != 0) {

yOffset = 0; // двигаем по горизонтали поэтому смещение обнуляем для вертикали

xOffset -= dx \* metrics.tileSize;

}

} else {

// движение по оси у

dy = (yOffset) / metrics.tileSize;

if (dy != 0) {

xOffset = 0; // двигаемся по вертикали смещение обнуляем для горизонтали

yOffset -= dy \* metrics.tileSize;

}

}

performMove(-dx, -dy); // производим само смещние

xTouch = (int) event.getX();

yTouch = (int) event.getY();

}

return true;

}

});

setOnKeyListener((v, keyCode, event) -> {

if (event.getAction() != KeyEvent.ACTION\_DOWN)

return false;

switch (keyCode) {

case KeyEvent.KEYCODE\_DPAD\_UP:

performMove(0, -1);

break;

case KeyEvent.KEYCODE\_DPAD\_RIGHT:

performMove(1, 0);

break;

case KeyEvent.KEYCODE\_DPAD\_DOWN:

performMove(0, 1);

break;

case KeyEvent.KEYCODE\_DPAD\_LEFT:

performMove(-1, 0);

break;

default:

return false;

}

return true;

});

}

//метод отмена хода вызываем из актиности

public void backPressed() {

if (game.performUndo()) {

centerScreenOnPlayerIfNecessary();

invalidate();

} else if (game.undos.isEmpty()) { //если дейсвтий нет то вызываем вибрацию

Vibrator vibrator = (Vibrator) getContext().getSystemService(Context.VIBRATOR\_SERVICE);

vibrator.vibrate(400);

}

}

//игрок по центру экрана

private void centerScreenOnPlayer() {

int[] playerPos = game.getPlayerPosition();

int centerX = playerPos[0] \* metrics.tileSize + metrics.tileSize / 2;

int centerY = playerPos[1] \* metrics.tileSize + metrics.tileSize / 2;

// центрируем поле

offsetX = centerX - getWidth() / 2;

offsetY = centerY - getHeight() / 2;

offsetX = -offsetX;

offsetY = -offsetY;

}

private void centerScreenOnPlayerIfNecessary() {

if (metrics.levelFitsOnScreen) { //если поле видно полностью то ничего не делаем

return;

}

int[] playerPos = game.getPlayerPosition();

int playerX = playerPos[0];

int playerY = playerPos[1];

int tileSize = metrics.tileSize; //размер плитки

int tilesLeftOfPlayer = (playerX \* tileSize + offsetX) / tileSize; //количество плиток слева

int tilesRightOfPlayer = (getWidth() - playerX \* tileSize - offsetX) / tileSize; //количество плиток справа

int tilesAboveOfPlayer = (playerY \* tileSize + offsetY) / tileSize;//количество плиток сверху

int tilesBelowOfPlayer = (getHeight() - playerY \* tileSize - offsetY) / tileSize; //количество плиток ниже игрока

int MINPOROG = 1; //минимум

if (tilesLeftOfPlayer <= MINPOROG ||

tilesRightOfPlayer <= MINPOROG ||

tilesAboveOfPlayer <= MINPOROG

|| tilesBelowOfPlayer <= MINPOROG) {

centerScreenOnPlayer();

ignoreDrag = true; // включаем флаг игнорирования перемещения игрок по центру экрана

}

}

private void computeMetrics(int newImageSize) {

metrics = new GameMetrics();

metrics.tileSize = newImageSize;

// проверка помещяется ли все поле на экране

metrics.levelFitsOnScreen = ((game.getWidthInTiles() - 1) \* metrics.tileSize <= getWidth() && (game

.getHeightInTiles() - 1) \* metrics.tileSize <= getHeight());

}

//меняем плитки под новый размер

public void customSizeChanged(int newImageSize) {

computeMetrics(newImageSize == 0 ? 64: newImageSize);

Resources resources = getResources();

BitmapFactory.Options options = new BitmapFactory.Options();

options.inScaled = false;

//считываем рисунки для плиток

boxOnFloorBitmap = BitmapFactory.decodeResource(resources, R.drawable.box\_on\_floor, options);

boxOnEndPointBitmap = BitmapFactory.decodeResource(resources, R.drawable.box\_on\_endpoint, options);

floorBitmap = BitmapFactory.decodeResource(resources, R.drawable.ground, options);

playerOnFloorBitmap = BitmapFactory.decodeResource(resources, R.drawable.player\_down, options);

playerOnTargetBitmap = BitmapFactory.decodeResource(resources, R.drawable.player\_up, options);

outsideBitmap = BitmapFactory.decodeResource(resources, R.drawable.grass, options);

targetBitmap = BitmapFactory.decodeResource(resources, R.drawable.endpoint, options);

wallBitmap = BitmapFactory.decodeResource(resources, R.drawable.wall, options);

float scaleFactor = metrics.tileSize / 64.0f;

Matrix matrix = new Matrix();

matrix.postScale(scaleFactor, scaleFactor);

int imageSize = 64;

// пересоздаем изображения с новым размером

boxOnFloorBitmap = Bitmap.createBitmap(boxOnFloorBitmap, 0, 0, imageSize, imageSize, matrix, true);

boxOnEndPointBitmap = Bitmap.createBitmap(boxOnEndPointBitmap, 0, 0, imageSize, imageSize, matrix, true);

floorBitmap = Bitmap.createBitmap(floorBitmap, 0, 0, imageSize, imageSize, matrix, true);

playerOnFloorBitmap = Bitmap.createBitmap(playerOnFloorBitmap, 0, 0, imageSize, imageSize, matrix, true);

playerOnTargetBitmap = Bitmap.createBitmap(playerOnTargetBitmap, 0, 0, imageSize, imageSize, matrix, true);

outsideBitmap = Bitmap.createBitmap(outsideBitmap, 0, 0, imageSize, imageSize, matrix, true);

targetBitmap = Bitmap.createBitmap(targetBitmap, 0, 0, imageSize, imageSize, matrix, true);

wallBitmap = Bitmap.createBitmap(wallBitmap, 0, 0, imageSize, imageSize, matrix, true);

if (metrics.levelFitsOnScreen) {

int w = game.getWidthInTiles() \* metrics.tileSize;

int h = game.getHeightInTiles() \* metrics.tileSize;

offsetX = (getWidth() - w) / 2;

offsetY = (getHeight() - h) / 2;

} else {

centerScreenOnPlayer();

}

}

//рисуем поле

@Override

public void draw(Canvas canvas) {

super.draw(canvas);

//заполняем фон

int x0 = 0;

int y0 = 0;

while (x0 < getWidth())

{

while(y0 < getHeight())

{

canvas.drawBitmap(outsideBitmap, x0, y0, null);// отображаем нашу картинку на выбранной позиции

y0 += metrics.tileSize;

}

y0 = 0;

x0 += metrics.tileSize;

}

if (isInEditMode())

return;

canvas.setDensity(Bitmap.DENSITY\_NONE);

//поличество плиток по ширине

final int widthInTiles = game.getWidthInTiles();

//кол-во плиток по высоте

final int heightInTiles = game.getHeightInTiles();

//разммер плитки

final int tileSize = metrics.tileSize;

for (int x = 0; x < widthInTiles; x++) {

for (int y = 0; y < heightInTiles; y++) {

//выставляем с учетом смещения под игрока

int left = offsetX + tileSize \* x;

int top = offsetY + tileSize \* y;

Bitmap tileBitmap;

char c = game.getItemAt(x, y);

switch (c) { //определяем символ карты и рисуем ту плитку какой символ выставлен на карте

case '\'':

tileBitmap = outsideBitmap;

break;

case GameState.CHAR\_WALL:

tileBitmap = wallBitmap;

break;

case GameState.CHAR\_PLAYER\_ON\_FLOOR:

tileBitmap = playerOnFloorBitmap;

break;

case GameState.CHAR\_PLAYER\_ON\_TARGET:

tileBitmap = playerOnTargetBitmap;

break;

case GameState.CHAR\_FLOOR:

tileBitmap = floorBitmap;

break;

case GameState.CHAR\_BOX\_ON\_FLOOR:

tileBitmap = boxOnFloorBitmap;

break;

case GameState.CHAR\_BOX\_ON\_ENDPOINT:

tileBitmap = boxOnEndPointBitmap;

break;

case GameState.CHAR\_TARGET:

tileBitmap = targetBitmap;

break;

default:

throw new IllegalArgumentException(String.format("Неизвестный символ в позиции (%d,%d): %c", x, y, c));

}

canvas.drawBitmap(tileBitmap, left, top, null);// отображаем нашу картинку на выбранной позиции

}

}

}

//завершение игры

void gameOver() {

//запускаем вибрацию

Vibrator vibrator = (Vibrator) getContext().getSystemService(Context.VIBRATOR\_SERVICE);

vibrator.vibrate(300);

//обновояем поле

invalidate();

SharedPreferences prefs = getContext().getSharedPreferences(ConstMain.SHARED\_PREFS\_NAME,

Context.MODE\_PRIVATE);

final String maxLevelPrefName = ConstMain.getMaxLevelPrefName(game.currentLevelSet);

int currentMaxLevel = prefs.getInt(maxLevelPrefName, 1);

int newMaxLevel = game.getCurrentLevel() + 2; //

String message = "уровень уже был пройдент - нового уровня не открыто!";

boolean levelSetDone = false;

//сохраняем фактп рохождения в хранилище

if (newMaxLevel > currentMaxLevel) {

if (newMaxLevel - 1 >= Levels.levelMaps.get(game.currentLevelSet).length) {

newMaxLevel--;

message = "Вы прошли все уровни! Поздравляем!";

levelSetDone = true; } else {

prefs.edit()

.putInt(maxLevelPrefName, newMaxLevel)

.commit();

message = "Новый уровень доступен!";

}

}

int levelDestination = newMaxLevel - 1;

boolean levelSetDoneFinal = levelSetDone;

//показываем диалог с поздравлением и возмодностью перехода к следующему уровню

new AlertDialog.Builder(getContext())

.setCancelable(false)

.setMessage(message)

.setTitle("Поздравляем")

.setPositiveButton("Продолжить?", (dialog, which) -> {

((Activity) getContext()).finish();

if (!levelSetDoneFinal) {

Intent intent = new Intent()

.putExtra(ConstMain.GAME\_LEVEL\_INTENT\_EXTRA, levelDestination)

.putExtra(ConstMain.GAME\_LEVEL\_SET\_EXTRA, game.currentLevelSet)

.setClass(getContext(), GameActivity.class);

getContext().startActivity(intent);

}

})

.show();

}

//вызов метода при изменении размерности

@Override

protected void onSizeChanged(int width, int height, int oldw, int oldh) {

super.onSizeChanged(width, height, oldw, oldh);

if (!isInEditMode())

customSizeChanged(0);

}

//перемещение

void performMove(int dx, int dy) {

if (game.tryMove(dx, dy)) {

centerScreenOnPlayerIfNecessary();

invalidate();

if (game.isDone()) {

gameOver(); // завершение игры

}

}

}

}

**Программный модуль ConstMain.java**

package game.me.socoban.constants;

//глобальный переменные

public class ConstMain {

public static final String GAME\_KEY = "GAME";

public static final String GAME\_LEVEL\_INTENT\_EXTRA = "GAME\_LEVEL";

public static final String GAME\_LEVEL\_SET\_EXTRA = "GAME\_LEVEL\_SET";

public static final String IMAGE\_SIZE\_PREFS\_KEY = "image\_size";

public static final String SHOW\_HELP\_INTENT\_EXTRA = "SHOW\_HELP";

public static final String MAX\_LEVEL\_NAME = "max\_level";

public static final String SHARED\_PREFS\_NAME = "game\_prefs";

public static String getMaxLevelPrefName(int levelSetIndex) {

return MAX\_LEVEL\_NAME + (levelSetIndex == 0 ? "" : ("\_" + levelSetIndex));

}

}

**Программный модуль GameActivity.java**

package game.me.socoban;

import android.app.Activity;

import android.app.AlertDialog;

import android.content.Intent;

import android.content.SharedPreferences;

import android.graphics.Point;

import android.os.Bundle;

import android.view.Display;

import android.view.View;

import game.me.socoban.constants.ConstMain;

import game.me.socoban.gamecontrol.GameState;

import game.me.socoban.gamecontrol.GameView;

public class GameActivity extends Activity {

public GameState gameState;

private int IMAGE\_SIZE; //размер плитки игрового поля

GameView gameView;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

//проверяем есть ли сохраненные данные о прогрессе

if (savedInstanceState != null) {

gameState = (GameState) savedInstanceState.getSerializable(ConstMain.GAME\_KEY); //получаем данные из сохраненнх значений

}

if (gameState == null) { //если нет сохранения то начальный уровень 0

int level = 0; //уровень

int levelSet = 0; //тип игры

Intent intent = getIntent();

if (intent != null && intent.getExtras() != null) {

level = intent.getExtras().getInt(ConstMain.GAME\_LEVEL\_INTENT\_EXTRA, 0);

if (intent.getExtras().getBoolean(ConstMain.SHOW\_HELP\_INTENT\_EXTRA, false))

showHelp(); // отображение информации о цели и управлении при первом показе

levelSet = intent.getExtras().getInt(ConstMain.GAME\_LEVEL\_SET\_EXTRA, 0);

}

gameState = new GameState(level, levelSet);

}

setContentView(R.layout.activity\_main);

//определяем размеры экрана

Display display = getWindowManager().getDefaultDisplay();

Point size = new Point();

display.getSize(size);

//вычисляем размерность наиболее подходящую для нашего экрана для поля 11 столбцов по умолчанию для 1 уровня

int defaultImageSize = Math.min(size.x, size.y) / 11;

if (defaultImageSize % 2 != 0) // если нечетное то делаем его четным

defaultImageSize--;

//выставляем значение размера из сохраненных значений (если их нет то наше расчитанное)

IMAGE\_SIZE = getSharedPreferences(ConstMain.SHARED\_PREFS\_NAME, MODE\_PRIVATE).getInt(ConstMain.IMAGE\_SIZE\_PREFS\_KEY,

defaultImageSize);

gameView = findViewById(R.id.game\_view); //наше поле игры

View undoBtn = findViewById(R.id.btn\_game\_undo); //кнопка ход назад

undoBtn.setOnClickListener(v -> gameView.backPressed()); //связываем короткое с действием игры ход назад

//на долгое нажатие назад делаем сброс уровня

undoBtn.setOnLongClickListener(v -> {

new AlertDialog

.Builder(this)

.setMessage(R.string.game\_restart)

.setPositiveButton(android.R.string.ok, (dialog, which) -> {

gameState.restart(); //перезапуск игры

gameView.invalidate(); //перерисовка игрового поля

})

.setNegativeButton(android.R.string.cancel, null).show();

return true;

});

View leaveBtn = findViewById(R.id.btn\_game\_leave); //кнопка покинуть уровень

leaveBtn.setOnClickListener(v -> finish()); //закрываем актиность с игрой

View viewPlusBtn = findViewById(R.id.btn\_view\_plus); //увеличить размер поля

viewPlusBtn.setOnClickListener(v -> changeViewSize(true)); //меняем размер поля

View viewMinusBtn = findViewById(R.id.btn\_view\_minus); // уменьшить размер поля

viewMinusBtn.setOnClickListener(v -> changeViewSize(false));//меняем размер поля

}

//первая часть процедуры изменения размер

void changeViewSize(boolean plus)

{

if (plus) //флаг увеличения

{

if (IMAGE\_SIZE < 136) //максимально у нас 136 размер плитки

setImageSize(IMAGE\_SIZE + 2); //по каждому клику увеличиваем на 2

}

else

{

if (IMAGE\_SIZE > 12) //минимальный 12

setImageSize(IMAGE\_SIZE - 2); //по каждому клику уменьшаю на 2

}

}

//изменяем размерность отображения игры (2 часть)

private void setImageSize(int newSize) {

IMAGE\_SIZE = newSize;

SharedPreferences prefs = getSharedPreferences(ConstMain.SHARED\_PREFS\_NAME, MODE\_PRIVATE);

prefs.edit()

.putInt(ConstMain.IMAGE\_SIZE\_PREFS\_KEY, newSize) //сохраняем значение в настройки

.commit();

gameView.customSizeChanged(newSize); //меняем значение на поле

gameView.invalidate(); // обвноляем поле

}

@Override

protected void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

outState.putSerializable(ConstMain.GAME\_KEY, gameState); //Сохраняем текущий прогресс прохождения

}

//выставляем окно на полный экран

@Override

public void onWindowFocusChanged(boolean hasFocus) {

super.onWindowFocusChanged(hasFocus);

if (hasFocus) {

getWindow().getDecorView().setSystemUiVisibility(

View.SYSTEM\_UI\_FLAG\_LAYOUT\_STABLE | View.SYSTEM\_UI\_FLAG\_LAYOUT\_HIDE\_NAVIGATION

| View.SYSTEM\_UI\_FLAG\_LAYOUT\_FULLSCREEN | View.SYSTEM\_UI\_FLAG\_HIDE\_NAVIGATION

| View.SYSTEM\_UI\_FLAG\_FULLSCREEN | View.SYSTEM\_UI\_FLAG\_IMMERSIVE\_STICKY);

}

}

//показ окна с текстом помощи

void showHelp() {

new AlertDialog.Builder(this)

.setMessage(

"Передвиньте ящики на указанные цели, чтобы пройти уровень(ящик изменит цвет). Проходите уровни, чтобы разблокировать новые.\n\n" +

"Увеличивайте или уменьшайте масштаб с помощью кнопок на экране. \n\n" +

"Отмена перемещений с помощью кнопки \"Отменить ход\"\n\n"+

"Перемещение с помощью касаний")

.setPositiveButton("Ok", null)

.create()

.show();

}

}

**Программный модуль MenuActivity.java**

package game.me.socoban;

import android.annotation.SuppressLint;

import android.app.Activity;

import android.app.AlertDialog;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import java.util.ArrayList;

import java.util.List;

import game.me.socoban.constants.ConstMain;

import game.me.socoban.constants.Levels;

/\*

класс описывающий экран с меню уровнями сложности игры

простой

средний

сложный

\*/

public class MenuActivity extends Activity {

@SuppressLint("NonConstantResourceId")

int getLvlfromBtn(View view)//в заивисмости от нажатой кнопки получаем индекс для типа

{

int index = -1;

switch (view.getId()) {

case R.id.btnLevelEasy:

index = 0;//легкий

break;

case R.id.btnLevelsMedium:

index = 1;//средний

break;

case R.id.btnLevelHard:

index = 2;//сложный

break;

}

return index;

}

//действия при нажатии на кнопку меню

public void onButtonClicked(View view) {

int levelSetIndex = getLvlfromBtn(view); // вид игры

SharedPreferences prefs = getSharedPreferences(ConstMain.SHARED\_PREFS\_NAME, MODE\_PRIVATE); //получаем экземпляр хранимых данных

String maxLevelNamePref = ConstMain.getMaxLevelPrefName(levelSetIndex);

int maxLevel = Math.min(prefs.getInt(maxLevelNamePref, 1),//узнаем максимальный уровень для типа игры

Levels.levelMaps.get(levelSetIndex).length); //по типу узнаем общее кол-во уровней

if (maxLevel == 1) { //если еще нет пройденных уровней то запускаем игру на начальном уровне для выбранного типа

Intent intent = new Intent()

.putExtra(ConstMain.GAME\_LEVEL\_INTENT\_EXTRA, 0)

.putExtra(ConstMain.GAME\_LEVEL\_SET\_EXTRA, levelSetIndex)

.putExtra(ConstMain.SHOW\_HELP\_INTENT\_EXTRA, true) //показывать охно помощи

.setClass(this, GameActivity.class);

startActivity(intent);

} else {

List<String> levelList = new ArrayList<String>(maxLevel);

for (int i = maxLevel; i > 0; i--) { //собираем спсиок пройденных уровней в список

levelList.add("Уровень " + i);

}

final String[] items = levelList.toArray(new String[maxLevel]);

new AlertDialog.Builder(this) //показываем окно с выбором уровня

.setTitle("Выберите уровень")

.setItems(items, (dialog, item) -> {

int levelClicked = maxLevel - item - 1;

Intent intent = new Intent()

.putExtra(ConstMain.GAME\_LEVEL\_SET\_EXTRA, levelSetIndex) //тип игры

.putExtra(ConstMain.GAME\_LEVEL\_INTENT\_EXTRA, levelClicked) //уровень

.setClass(this, GameActivity.class);

startActivity(intent);

})

.create()

.show();

}

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_levelsmenu); //биндим активность к макету

}

@Override

protected void onResume() {

super.onResume();

/\*описываем изменение текста после возврата в меню - текущий прогресс прохождения в сложностях\*/

setButtonText(R.id.btnLevelEasy, 0);

setButtonText(R.id.btnLevelsMedium, 1);

setButtonText(R.id.btnLevelHard, 2);

}

//меняем текст для кнопки меню указывая прогресс прохождения

private void setButtonText(int buttonId, int levelSetIndex) {

Button button = findViewById(buttonId); //находим кнопку

String buttonText = button.getText().toString(); //поулчаем текущий текст

if (buttonText.contains("-")) {

buttonText = buttonText.split("-")[0].trim(); //делим текст на уровень и текст по разделителю

}

SharedPreferences prefs = getSharedPreferences(ConstMain.SHARED\_PREFS\_NAME, MODE\_PRIVATE);

String maxLevelNamePref = ConstMain.getMaxLevelPrefName(levelSetIndex);

int maxLevel = Math.min(prefs.getInt(maxLevelNamePref, 1), Levels.levelMaps.get(levelSetIndex).length);

int availableLevels = Levels.levelMaps.get(levelSetIndex).length; //доступные уровни

button.setText(String.format("%s - %d/%d", buttonText, maxLevel, availableLevels)); //выставляем новый тест тип игры максимальный пройденный уровень - всего уровней

}

}

**Программный модуль Levels.java**

package game.me.socoban.constants;

import java.util.ArrayList;

import java.util.List;

/\*

синглтон содержащий описание уровней для игры в сивольном формате

# - стена

. - цель положения

@ - игрок

$ - наш ящик

\*/

public class Levels {

public static final List<String[][]> levelMaps = new ArrayList<String[][]>();

//заполняем массив

static {

// Простые

levelMaps.add(new String[][] {

{

"####''"

,

"# .#''"

,

"# ###"

,

"#\*@ #"

,

"# $ #"

,

"# ###"

,

"####''"

}

,

{

"######"

,

"# #"

,

"# #@ #"

,

"# $\* #"

,

"# .\* #"

,

"# #"

,

"######"

}

,

{

"''####'''"

,

"### ####"

,

"# $ #"

,

"# # #$ #"

,

"# . .#@ #"

,

"#########"

}

,

{

"########"

,

"# #"

,

"# .\*\*$@#"

,

"# #"

,

"##### #"

,

"''''####"

}

,

{

"'#######"

,

"'# #"

,

"'# .$. #"

,

"## $@$ #"

,

"# .$. #"

,

"# #"

,

"########"

}

,

{

"######'#####"

,

"# ### #"

,

"# $$ #@#"

,

"# $ #... #"

,

"# ########"

,

"#####'''''''"

}

,

{

"#######"

,

"# #"

,

"# .$. #"

,

"# $.$ #"

,

"# .$. #"

,

"# $.$ #"

,

"# @ #"

,

"#######"

}

,

{

"''######"

,

"''# ..@#"

,

"''# $$ #"

,

"''## ###"

,

"'''# #''"

,

"'''# #''"

,

"#### #''"

,

"# ##'"

,

"# # #'"

,

"# # #'"

,

"### #'"

,

"''#####'"

}

,

{

"#####'"

,

"#. ##"

,

"#@$$ #"

,

"## #"

,

"'## #"

,

"''##.#"

,

"'''###"

}

,

{

"''''''#####"

,

"''''''#. #"

,

"''''''#.# #"

,

"#######.# #"

,

"# @ $ $ $ #"

,

"# # # # ###"

,

"# #''"

,

"#########''"

}

,

{

"''######'"

,

"''# #'"

,

"''# ##@##"

,

"### # $ #"

,

"# ..# $ #"

,

"# #"

,

"# ######"

,

"####'''''"

}

,

{

"#######"

,

"# #"

,

"#. . #"

,

"# ## ##"

,

"# $ #'"

,

"###$ #'"

,

"''#@ #'"

,

"''# #'"

,

"''####'"

}

,

{

"########"

,

"# .. #"

,

"# @$$ #"

,

"##### ##"

,

"'''# #'"

,

"'''# #'"

,

"'''# #'"

,

"'''####'"

}

,

{

"#######''"

,

"# ###"

,

"# @$$..#"

,

"#### ## #"

,

"''# #"

,

"''# ####"

,

"''# #'''"

,

"''####'''"

}

,

{

"####'''"

,

"# ####"

,

"# . . #"

,

"# $$#@#"

,

"## #"

,

"'######"

}

,

{

"#####''"

,

"# ###"

,

"#. . #"

,

"# # #"

,

"## # #"

,

"'#@$$ #"

,

"'# #"

,

"'# ###"

,

"'####''"

}

,

{

"#######"

,

"# \* #"

,

"# #"

,

"## # ##"

,

"'#$@.#'"

,

"'# #'"

,

"'#####'"

}

,

{

"#'#####"

,

"''# #"

,

"###$$@#"

,

"# ###"

,

"# #"

,

"# . . #"

,

"#######"

}

,

{

"'####''"

,

"'# ###"

,

"'# $$ #"

,

"##... #"

,

"# @$ #"

,

"# ###"

,

"#####''"

}

,

{

"'#####"

,

"'# @ #"

,

"'# #"

,

"###$ #"

,

"# ...#"

,

"# $$ #"

,

"### #"

,

"''####"

}

,

{

"######'"

,

"# .#'"

,

"# ## ##"

,

"# $$@#"

,

"# # #"

,

"#. ###"

,

"#####''"

}

,

{

"#####''"

,

"# #''"

,

"# @ #''"

,

"# $$###"

,

"##. . #"

,

"'# #"

,

"'######"

}

,

{

"'''''#####'"

,

"'''''# ##"

,

"'''''# #"

,

"'###### #"

,

"## #. #"

,

"# $ $ @ ##"

,

"# ######.#'"

,

"# #'"

,

"##########'"

}

,

{

"####''"

,

"# ###"

,

"# $$ #"

,

"#... #"

,

"# @$ #"

,

"# ##"

,

"#####'"

}

,

{

"''####'"

,

"'## #'"

,

"##@$.##"

,

"# $$ #"

,

"# . . #"

,

"### #"

,

"''#####"

}

,

{

"'####''"

,

"## ###"

,

"# #"

,

"#.\*\*$@#"

,

"# ###"

,

"## #''"

,

"'####''"

}

,

{

"#######"

,

"#. # #"

,

"# $ #"

,

"#. $#@#"

,

"# $ #"

,

"#. # #"

,

"#######"

}

,

{

"''####'''"

,

"### ####"

,

"# #"

,

"#@$\*\*\*. #"

,

"# #"

,

"#########"

}

,

{

"''####'"

,

"'## #'"

,

"'#. $#'"

,

"'#.$ #'"

,

"'#.$ #'"

,

"'#.$ #'"

,

"'#. $##"

,

"'# @#"

,

"'## #"

,

"''#####"

}

,

{

"####'''''''''''"

,

"# ############"

,

"# $ $ $ $ $ @ #"

,

"# ..... #"

,

"###############"

}

,

{

"''''''###"

,

"#####'#.#"

,

"# ###.#"

,

"# $ #.#"

,

"# $ $ #"

,

"#####@# #"

,

"''''# #"

,

"''''#####"

}

});

// Средние

levelMaps.add(new String[][] {

{

"'''###'''''"

,

"''## #'####"

,

"'## ### #"

,

"## $ #"

,

"# @$ # #"

,

"### $### #"

,

"''# #.. #"

,

"'## ##.# ##"

,

"'# ##'"

,

"'# ##''"

,

"'#######'''"

}

,

{

"'##'#####"

,

"##'## . #"

,

"#'## $. #"

,

"'## $ #"

,

"## $@ ###"

,

"# $ ##''"

,

"#.. ##'##"

,

"# #'##'"

,

"#####'#''"

}

,

{

"'''''''''''#####"

,

"''''''''''## #"

,

"''''''''''# #"

,

"''''####''# $ ##"

,

"''''# ####$ $#'"

,

"''''# $ $ #'"

,

"'''## ## $ $ $#'"

,

"'''# .# $ $ #'"

,

"'''# .# #'"

,

"##### #########'"

,

"#.... @ #''''''"

,

"#.... #''''''"

,

"## ######''''''"

,

"'####'''''''''''"

}

,

{

"''###########"

,

"'## # @#"

,

"### $ $$# #"

,

"#'##$ $$ #"

,

"#''# $ # #"

,

"###### ######"

,

"#.. ..$ #\*##'"

,

"# .. ###''"

,

"# ..#####'''"

,

"#########''''"

}

,

{

"''###########''"

,

"'## # ##'"

,

"### $ $#$ $ ###"

,

"# #$ $ # $ $# #"

,

"# $ ..#.. $ #"

,

"# $...#...$ #"

,

"# $ .. \* .. $ #"

,

"###### @ ######"

,

"# $ .. .. $ #"

,

"# $...#...$ #"

,

"# $ ..#.. $ #"

,

"# #$ $ # $ $# #"

,

"### $ $#$ $ ###"

,

"'## # ##'"

,

"''###########''"

}

,

{

"''###########''"

,

"###. .$. .###"

,

"'## $ $ $ ##'"

,

"''## ..$.. ##''"

,

"'''##$#$#$##'''"

,

"''''#.$ $.#''''"

,

"''''# @ #''''"

,

"''''### ###''''"

,

"'''## $ $ ##'''"

,

"'''#. $ .#'''"

,

"'''### . ###'''"

,

"'''''#####'''''"

}

,

{

"'''''''''''######''"

,

"''''####''## #''"

,

"''### #''# ## ###"

,

"### #### # $ #"

,

"# $ @ ...\*.. $ #"

,

"# $ $ ## ### ###"

,

"### ### #'#####''"

,

"'# ###''''''''"

,

"'# ####''''''''''"

,

"'#####'''''''''''''"

}

,

{

"''''#######''"

,

"''''# ##'"

,

"##### ### ##"

,

"# # ##"

,

"#@$\*\*\*. ##$ #"

,

"# # ## .#"

,

"## ## # $ #"

,

"'## ####.$.#"

,

"''## #"

,

"'''###### ##"

,

"''''''''####'"

}

,

{

"#########"

,

"#. . #"

,

"#.$. . #"

,

"## ###@ #"

,

"'# $ ##"

,

"'# $$ ##'"

,

"'# $ #''"

,

"'# ###''"

,

"'####''''"

}

,

{

"''''''######''''''"

,

"''''''# #''''''"

,

"''''''# @ ###''''"

,

"####''# #''''"

,

"# ####..#.#$#####"

,

"# $ $ ##... #"

,

"# .....#$$ #"

,

"###### ##$## #####"

,

"'''''# $ #''''"

,

"'''''#### ####''''"

,

"'''''''# #'''''''"

,

"'''''''# #####'''"

,

"'''''### $ #'''"

,

"'''''# $ $ #'''"

,

"'''''# #$# ####'''"

,

"'''''# #''''''"

,

"'''''#######''''''"

}

,

{

"''####'''''"

,

"### ####''"

,

"# @ ##'"

,

"# #. .#.###"

,

"# $$$ $$$ #"

,

"###.#.#.# #"

,

"'## #"

,

"''#### ###"

,

"'''''####''"

}

,

{

"''''''''''''''#####''''''''"

,

"''''''''''''''# #''''''''"

,

"''''#####'''''# #######''"

,

"''''# #####'# ..... #''"

,

"##### # ## #'# # # #''"

,

"# $ $ $ $ $ #'## ## $ #''"

,

"# # ##......#### ### $$ ###"

,

"# ## \* # #'# $$ #"

,

"##########+$$ ##'# #"

,

"'''''''''#.$ $# #''########"

,

"'''''''''#.## #''''''''''"

,

"'''''''''########''''''''''"

}

,

{

"'''#######''"

,

"'### ##'"

,

"'# ### #'"

,

"'# # #'"

,

"###$#@ # #'"

,

"# ##### #'"

,

"# # \*. #'"

,

"##$$# \*.##'"

,

"'# \*..#'"

,

"'#### #...##"

,

"''''# #$$$ #"

,

"''''# $ #"

,

"''''##### #"

,

"''''''''####"

}

,

{

"'#######''"

,

"## #'#"

,

"# \*.$.#''"

,

"# \*.#.###"

,

"# #$@$$ #"

,

"# ## # #"

,

"###### #"

,

"'''''#####"

}

,

{

"'''#'#'#'#'#'#'#'#'#'#'#'#''"

,

"''# # # # # # # # # # # # #'"

,

"'# .$ . $. . $ .$ #"

,

"''# $# #$# # # #$# # # # #'"

,

"'# . . $. . .$ . . #"

,

"''# #$# # # # #$#$ $# #$# #'"

,

"'# . .$ .$ . $. . #"

,

"''# $# # # # #@# # # # #$ #'"

,

"'# . .$ . $. $. . #"

,

"''# #$# #$ $#$# # # # #$# #'"

,

"'# . . $. . .$ . . #"

,

"''# # # # #$# # # #$# #$ #'"

,

"'# $. $ . .$ . $. #"

,

"''# # # # # # # # # # # # #'"

,

"'''#'#'#'#'#'#'#'#'#'#'#'#''"

}

});

// Сложные

levelMaps.add(new String[][] {

{

"#####'''''"

,

"# #####'"

,

"# $ $ $ #'"

,

"### # # #'"

,

"''# # #'"

,

"'## ### ##"

,

"'# .....@#"

,

"'# $ $ #"

,

"'# ### ###"

,

"'# #''"

,

"'#######''"

}

,

{

"'''''####'"

,

"######..#'"

,

"# . #'"

,

"# # ..#'"

,

"# ## ###'"

,

"# $ ##"

,

"# # #$ @#"

,

"# # $ $ #"

,

"# ## $ ##"

,

"##### #'"

,

"''''#####'"

}

,

{

"''''''''''#####''"

,

"'''''''''## ##'"

,

"'''''''''# ##"

,

"'''''''''# @ #"

,

"############ #. #"

,

"# #.##"

,

"# ############.#'"

,

"# .#'"

,

"##$#$#$#$#$#$#.#'"

,

"'# .#'"

,

"'###############'"

}

,

{

"'''''''#####''''"

,

"'''''### #''''"

,

"####'# $$ ####'"

,

"# ### $ # #'"

,

"# ###$$ $ #'"

,

"# \*# @ ## # #'"

,

"## ##### #..# ##"

,

"'# ## #..# #"

,

"'## $ # #.. #"

,

"#'## $ # ..####"

,

"'#'## # #'''"

,

"''#'#### ##'''"

,

"'''#'''#####''''"

}

,

{

"'''''#####''"

,

"''#### #''"

,

"''# . # ###"

,

"''# $. $ $ #"

,

"''# #.## $ #"

,

"###@#..#$ #"

,

"# ##.# ##"

,

"# $ $ .# $#'"

,

"## ###.## #'"

,

"'# #'"

,

"'##### ###'"

,

"'''''####'''"

}

,

{

"''''#####''''''''''"

,

"'''## ######'''''"

,

"'''# @ # #'''''"

,

"'''# # $ \* #'''''"

,

"#### ###$#. #'''''"

,

"#.....# .##'''''"

,

"#.....# #$#. ###'''"

,

"### . ## #'''"

,

"'# ## ##$ ### ####"

,

"'# # # $ $ #"

,

"'# $$$# # $ $ #"

,

"'##### # ###### #"

,

"'''''#$ #'''####"

,

"''''## $ # #'''''''"

,

"''''# $ #'''''''"

,

"''''# ###'''''''"

,

"''''######'''''''''"

}

,

{

"'''''''''#####''''"

,

"########## #''''"

,

"#. ........ .##'''"

,

"# #### # ##''"

,

"## $ # # ##'"

,

"'# $ #'# # ##"

,

"'# $ $ # # # #"

,

"'# $ $ # @ #"

,

"'# $ $$ # #"

,

"'# $## ###########"

,

"'# # #''''''''''"

,

"'# ##''''''''''"

,

"'######'''''''''''"

}

,

{

"'''''#########''"

,

"'##### #''"

,

"## #### #''"

,

"# $ # @ \*..\*###"

,

"# # # #.... #"

,

"# #$# #.... #"

,

"# $# # ##$### #"

,

"# #$ ##"

,

"## $ $# ####'"

,

"'## $ ####''''"

,

"''### $$ #'''#''"

,

"''''# #'###''"

,

"''''######''''''"

}

,

{

"''''''''''''''####'''''''''''"

,

"'''''''''''#### ##'###''''''"

,

"''''''''#### $ #'#\*#'####'"

,

"'######## $ $ #'###'# #'"

,

"'# $ $ $ $ ## #'''''# ##"

,

"## $ $ #### $ ####### #"

,

"# $$ ##### $ ## # #"

,

"# ## ## $ $ # #"

,

"#.# $ # $ $ #### ## ## #"

,

"#.# $ $ $ #### # #"

,

"#.#.# $ # # #"

,

"#.#. ### # ######## #"

,

"#.#.###@#### ## # ##"

,

"#............. # ###### #'"

,

"# .########### ## #''''####'"

,

"#####'''''''''# #'''''''''"

,

"''''''''''''''######'''''''''"

}

,

{

"'''''''''''''####'''"

,

"'''#####'##### ##''"

,

"''## ### ##'"

,

"'## \* \* . $ # @ ##"

,

"'# ## \* ## ### # #"

,

"'# ## \* # # $ # #"

,

"'# # \* # # # #"

,

"## # ## # # ### #"

,

"# ## #'#. $ .## ##"

,

"# #.# ##'## #.# ##'"

,

"# # $ #. $ .$ ##''"

,

"# #\* $.# . ##'''"

,

"##### ## ## # ##''''"

,

"''''# # $ # #'''''"

,

"''''# # # ##'''''"

,

"''''# #$## #''''''"

,

"''''## ##''''''"

,

"'''''########'''''''"

}

,

{

"''''####''''''''''''"

,

"''''# ####'''''''''"

,

"''''# $ #########'"

,

"''''# .# $ ## #'"

,

"''''# $# .## $ ##"

,

"'#### .### #$$ #"

,

"## ## # .. # $$ #"

,

"# $ ...# $ #"

,

"# $ #####... # ##"

,

"# $# # .\*\*@####'"

,

"### # # # #''''"

,

"''##### #### #'#''"

,

"''''''## ##''''"

,

"'''''''########'''''"

}

,

{

"#''###'''''"

,

"'## ##'''"

,

"'#\*.$ #'''"

,

"# .$.$ .##'"

,

"# $.$.$ #'"

,

"# $.@.$ #"

,

"'# $.$.$ #"

,

"'##. $.$. #"

,

"'''# $.\*#'"

,

"'''## ##'"

,

"'''''###''#"

}

,

{

"''''#####''''''''"

,

"''''# #####''''"

,

"''''# # # #''''"

,

"''''# # #''''"

,

"'#####.# ...#####"

,

"'# .$$ ###$# #"

,

"'# #.# $. # #"

,

"'# .# $$$ # #"

,

"### # $@$ # ###"

,

"# # $$$ #. #'"

,

"# # .$ #.# #'"

,

"# #$### $$. #'"

,

"#####... #.#####'"

,

"''''# # #''''"

,

"''''# # # #''''"

,

"''''##### #''''"

,

"''''''''#####''''"

}

,

{

"''''#######'''"

,

"''''# ###'"

,

"''''# ###$ ##"

,

"''''#....$ #"

,

"''### ## # #"

,

"###@.$ #'# ##"

,

"# .\*.$ ## #'"

,

"# $.$ #.$$#'"

,

"# ## ## #'"

,

"##### # ###'"

,

"''''### $ #''"

,

"''''''## #''"

,

"'''''''#####''"

}

,

{

"''#####''''''''''''''''''''"

,

"''# #####''''''''''''''''"

,

"''# $$# #####''''''''''''"

,

"''# . $ # #####''''''''"

,

"'### ## . $ # #####''''"

,

"'# ##.### ....$ # #####"

,

"'# $$# ###.##. # $ # #"

,

"'# # $ # .##.### .$$ #"

,

"### ## .$$$# ###.## #"

,

"# ##.### #$$$. ## ###"

,

"# $$. ###.##. # $ # #'"

,

"# # $ # .##.### #$$ #'"

,

"##### # $.... ###.## #'"

,

"''''##### # $ . ## ###'"

,

"''''''''##### # $ . #''"

,

"''''''''''''##### #$$ #''"

,

"''''''''''''''''##### @#''"

,

"''''''''''''''''''''#####''"

}

});

}

}