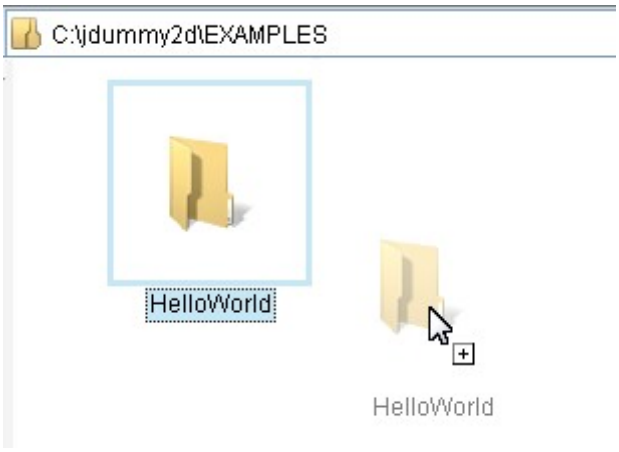
BalloonShooter jdummy2d tutorial

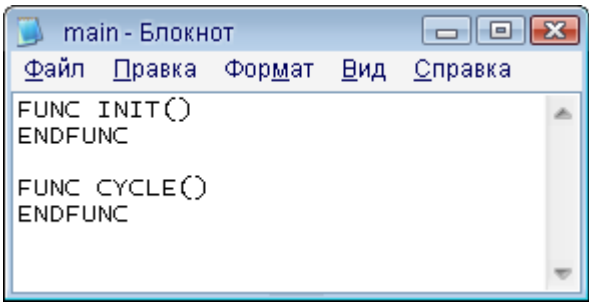
Сделаем на jdummy2d игру про ловлю шариков. Шарики летят вверх, надо успевать нажимать их, чтобы лопать. Пропустили – проиграли.

Для начала создаём проект.

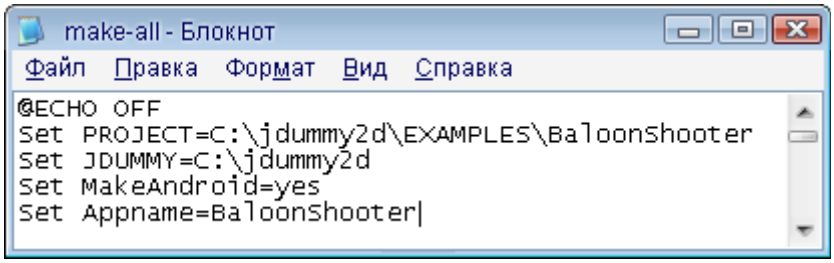


Копируем папку с примером и переименовываем её в *BalloonShooter*

В *sources/main.txt* убираем всё лишнее, остаётся только:



Открываем *make-all.bat* и *make-desktop.bat*, правим пути, имя:

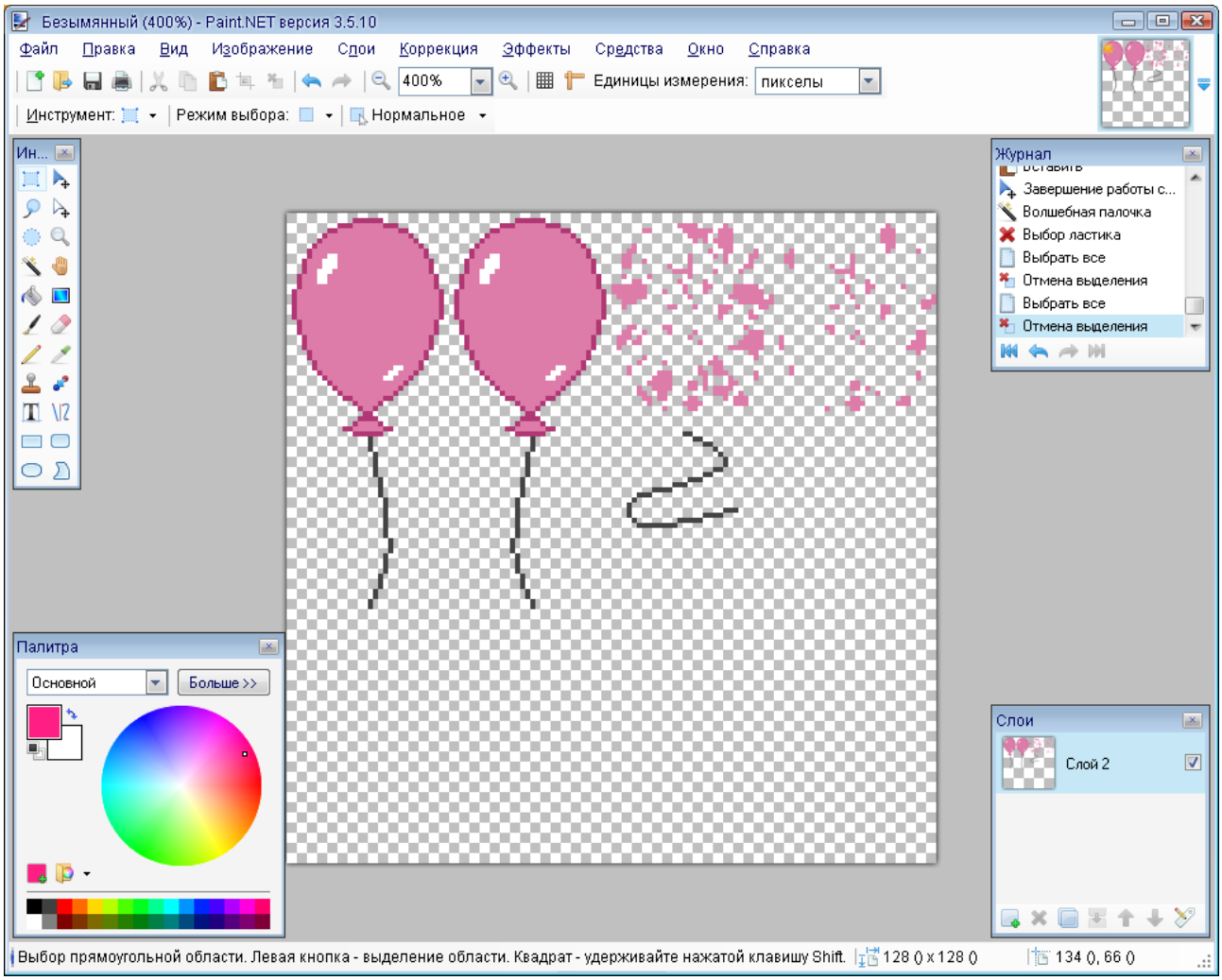


(в *make-desktop.bat*: MakeAndroid=no)

Копируем эти 4 строчки (начинающиеся на Set) в *android-install.bat* и *android-uninstall.bat*. Остальное не трогаем.

Новый проект создан.

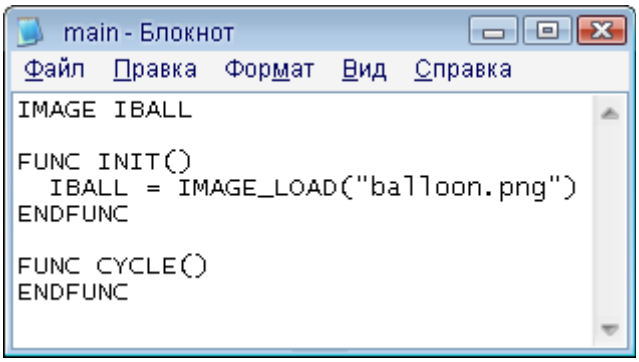
Нарисуем графику. Её будет не много. Спрайт-лист шарика получился размером 128x128
(в 64 по высоте не уместился).



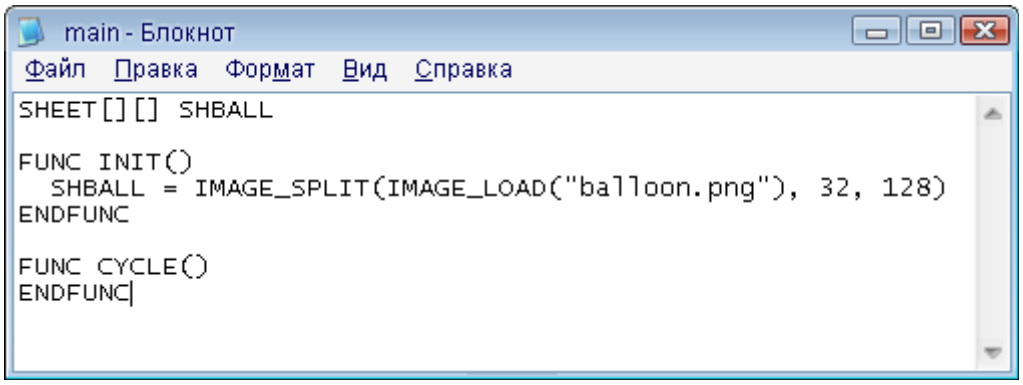
Я нарисовал всё в MsPaint из Windows XP, а в Paint.NET я только подчистил фон.

Сохраняем его в папку *data/assets/* нашего проекта как *balloon.png*

Сразу же добавим balloon.png в код как IMAGE:



Поскольку это спрайт-лист, нам надо создать спрайт-лист и разрезать изображение. Заметьте, что я убрал объявление IBALL, поскольку целое изображение нам не понадобится. Можно было бы оставить переменную IBALL и написать:
SHBALL = IMAGE_SPLIT(IBALL,32,128), но я сделал так:



Теперь SHBALL[][] содержит в ячейках [0][0], [1][0], [2][0] и [3][0] кусочки нашего изображения.

У нас будет много шаров. Хранить их позиции можно в векторах. А можно создать массив спрайтов, у которых есть свои позиции. 16 спрайтов должно хватить всем:

```
SHEET[][] SHBALL
SPRITE[] BALL

FUNC INIT()
  SHBALL = IMAGE_SPLIT(IMAGE_LOAD("balloon.png"), 32, 128)
  BALL = NEW SPRITE[16]
  // назначаем каждому спрайту первый кадр спрайт-листа:
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I] = NEW SPRITE(SHBALL[0][0])
  NEXT
ENDFUNC

FUNC CYCLE()
ENDFUNC
```

Тут же можно назначить случайные позиции шарам. Поскольку шары будут вылетать снизу, эти позиции будут сдвинуты на высоту экрана вниз.

```
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I] = NEW SPRITE(SHBALL[0][0])
    INT NEWX = RND(GETW()-32) -GETW()/2
    INT NEWY = RND(GETH()) -GETH()/2
    BALL[I].SETPOS(NEWX, NEWY-GETH())
  NEXT
```

Наконец, можно перейти к отображению этих спрайтов. Для этого заполним функцию CYCLE:

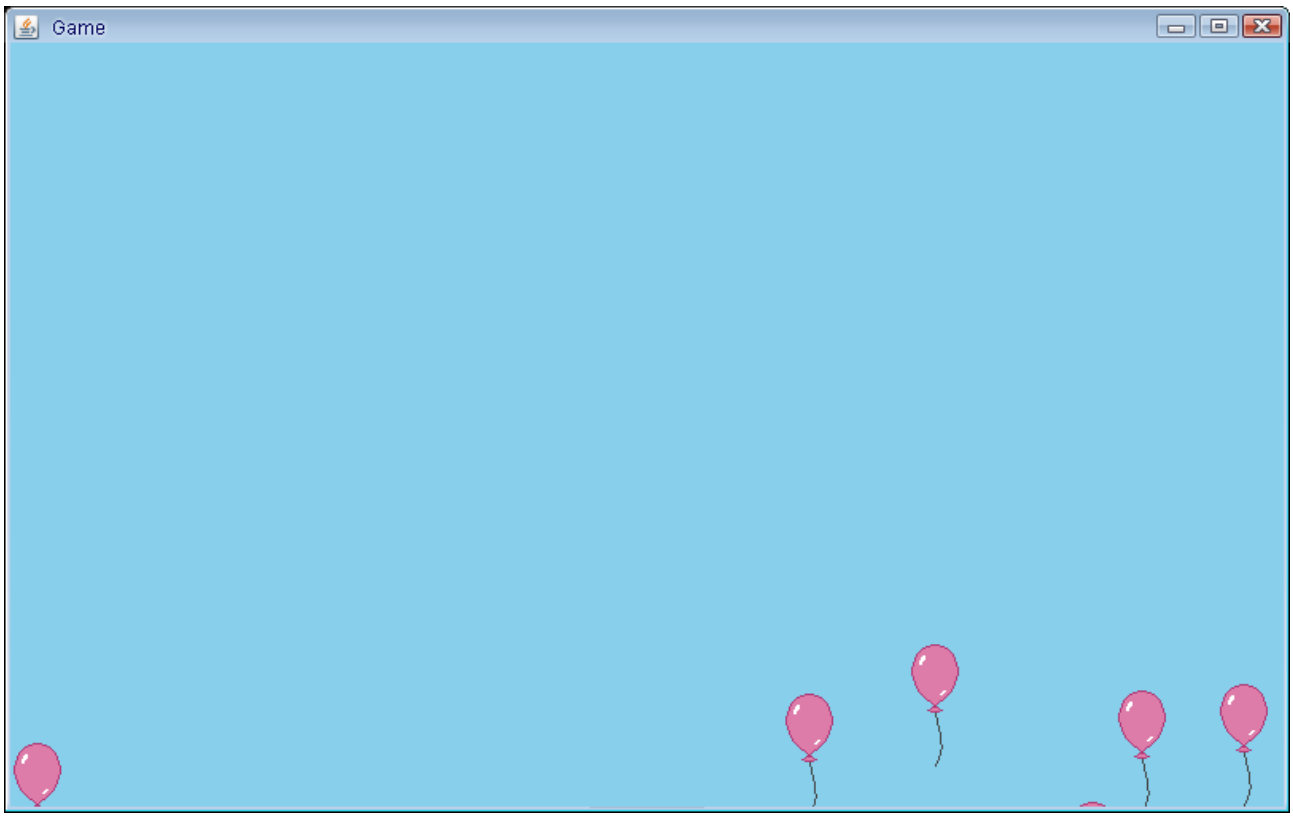
```
FUNC CYCLE()
  // очищаем экран цветом неба
  CLS(SKY)
  // объявляем режим рисования изображений
  BEGIN_SPRITES()
  // проходим по всему массиву спрайтов
  FOR (INT I=0; I<BALL.LENGTH; I++)
    // и рисуем каждый
    BALL[I].DRAW()
  NEXT
  // объявляем конец режима рисования изображений
  END_SPRITES()
ENDFUNC
```

Полный листинг программы теперь выглядит так:

```
SHEET[][] SHBALL
SPRITE[] BALL

FUNC INIT()
  SHBALL = IMAGE_SPLIT(IMAGE_LOAD("balloon.png"), 32, 128)
  BALL = NEW SPRITE[16]
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I] = NEW SPRITE(SHBALL[0][0])
    INT NEWX = RND(GETW()-32) -GETW()/2
    INT NEWY = RND(GETH()) -GETH()/2
    BALL[I].SETPOS(NEWX, NEWY-GETH())
  NEXT
ENDFUNC

FUNC CYCLE()
  CLS(SKY)
  BEGIN_SPRITES()
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I].DRAW()
  NEXT
  END_SPRITES()
ENDFUNC
```

Запустите make-desktop.bat. Ошибок быть не должно (в этом уроке используется версия (jdummy2d-22aug2012). Запустите run-console.bat или двойной щелчок левой кнопкой по game.jar.

Должно получиться что-то вроде:



Шарики пока статичны.
Заставим их двигаться, напишем функцию MOVEBALLS и добавим её в конец нашей программы:

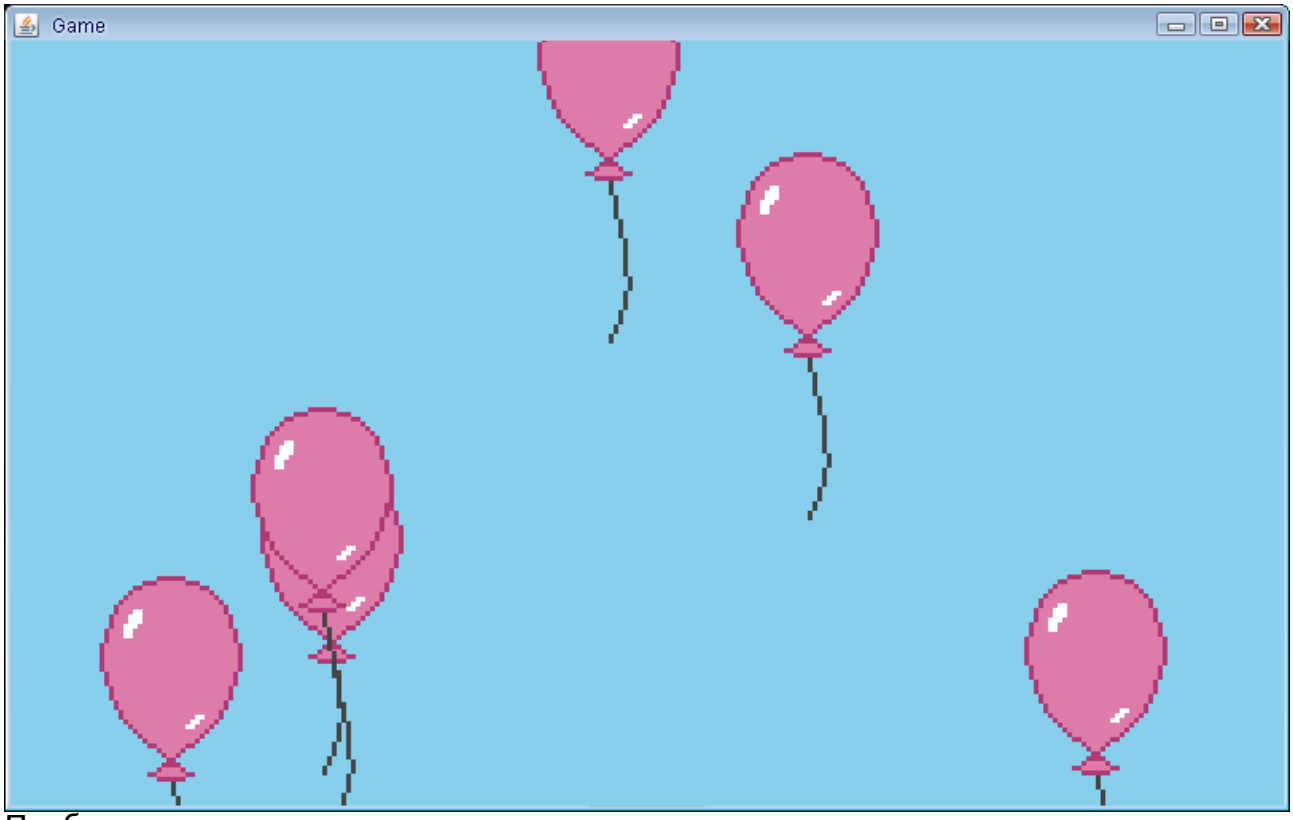
```
FUNC MOVEBALLS()
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I].MOVE(0,GETDELTA()*20)
  NEXT
ENDFUNC
```

Вынесем из функции CYCLE всё в функцию RENDER:

```
FUNC CYCLE()
  RENDER()
  MOVEBALLS()
ENDFUNC

FUNC RENDER()
  CLS(SKY)
  BEGIN_SPRITES()
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I].DRAW()
  NEXT
  END_SPRITES()
ENDFUNC

FUNC MOVEBALLS()
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I].MOVE(0,GETDELTA()*200)
  NEXT
ENDFUNC
```

После этих изменений запустите снова make-desktop.bat, затем game.jar. Мы видим, что шарики двигаются, но они слишком малы для экрана и их слишком много. Изменим функцию INIT:

```
FUNC INIT()
  SHBALL = IMAGE_SPLIT(IMAGE_LOAD("balloon.png"), 32, 128)
  BALL = NEW SPRITE[6]
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I] = NEW SPRITE(SHBALL[0][0])
    INT NEWX = RND(GETW()-64) -GETW()/2
    INT NEWY = RND(GETH()) -GETH()/2
    BALL[I].SETPOS(NEWX, NEWY-GETH())
    BALL[I].SCALE(2)
  NEXT
ENDFUNC
```



Проблема – некоторые шарики при генерации слишком сильно прячутся друг за друга.
Решим её так:

```
FUNC INIT()
  SHBALL = IMAGE_SPLIT(IMAGE_LOAD("balloon.png"), 32, 128)
  BALL = NEW SPRITE[6]
  INT OFFSET = 0
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I] = NEW SPRITE(SHBALL[0][0])
    INT NEWX = RND(GETW()/64)*64+64 -GETW()/2
    INT NEWY = -GETH()/2 - OFFSET
    BALL[I].SETPOS(NEWX, NEWY-GETH())
    BALL[I].SCALE(2)
    OFFSET += 64
  NEXT
```

```
ENDFUNC
```

Наши шарики всё еще без анимации. Добавим функцию ANIMATE. Жирным выделены новые переменные:

```
FUNC ANIMATE()
  // увеличиваем счетчик анимации
  ANIMCOUNTER += GETDELTA()*100
  IF (ANIMCOUNTER > ANIMSPEED) THEN
    // обнуляем счетчик
    ANIMCOUNTER = 0
  ELSE
    // если не насчитали 30, покидаем функцию
    RETURN
  ENDIF
  // назначаем случайные кадры спрайтам (0 или 1)
  FOR (INT I=0; I<BALL.LENGTH; I++)
    INT RF = RND(2)
    BALL[I].ASSIGN(SHBALL[RF][0])
  NEXT
ENDFUNC
```

В начало программы добавляем новые переменные:

```
FLOAT ANIMCOUNTER = 0, ANIMSPEED = 30
```

Изменившаяся функция *CYCLE:*

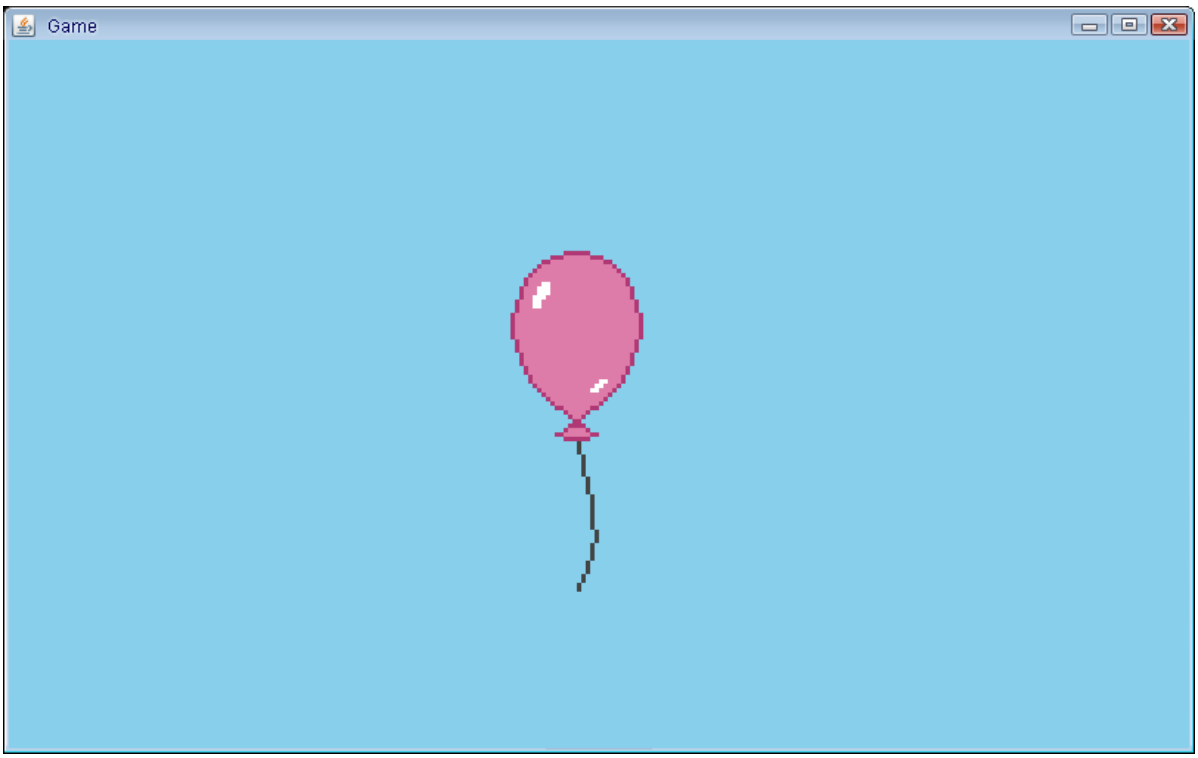
```
FUNC CYCLE()
  RENDER()
  MOVEBALLS()
  ANIMATE()
ENDFUNC
```

Начнем лопать шарики. Копируем из README функцию обработки события нажатия:

```
// Нажатие кнопки мыши или пальца на экран:
FUNC TOUCHDOWN(INT x, INT y, INT ptr, INT btn) RETURNS BOOL
     RETURN TRUE
ENDFUNC
```

Чтобы понять как соотносятся координаты позиции курсора и спрайта, попробуйте поиграть с этой функцией:

```
FUNC TOUCHDOWN(INT x, INT y, INT ptr, INT btn) RETURNS BOOL
  x -= GETW()/2
  y = -y
  y += GETH()/2
  BALL[0].SETPOS(x,y)
  RETURN TRUE
ENDFUNC
```



После нехитрых манипуляций находим нужное смещение до центра шарика:

```
x -= GETW()/2 + 24
y = -y
y += GETH()/2 – 212
```

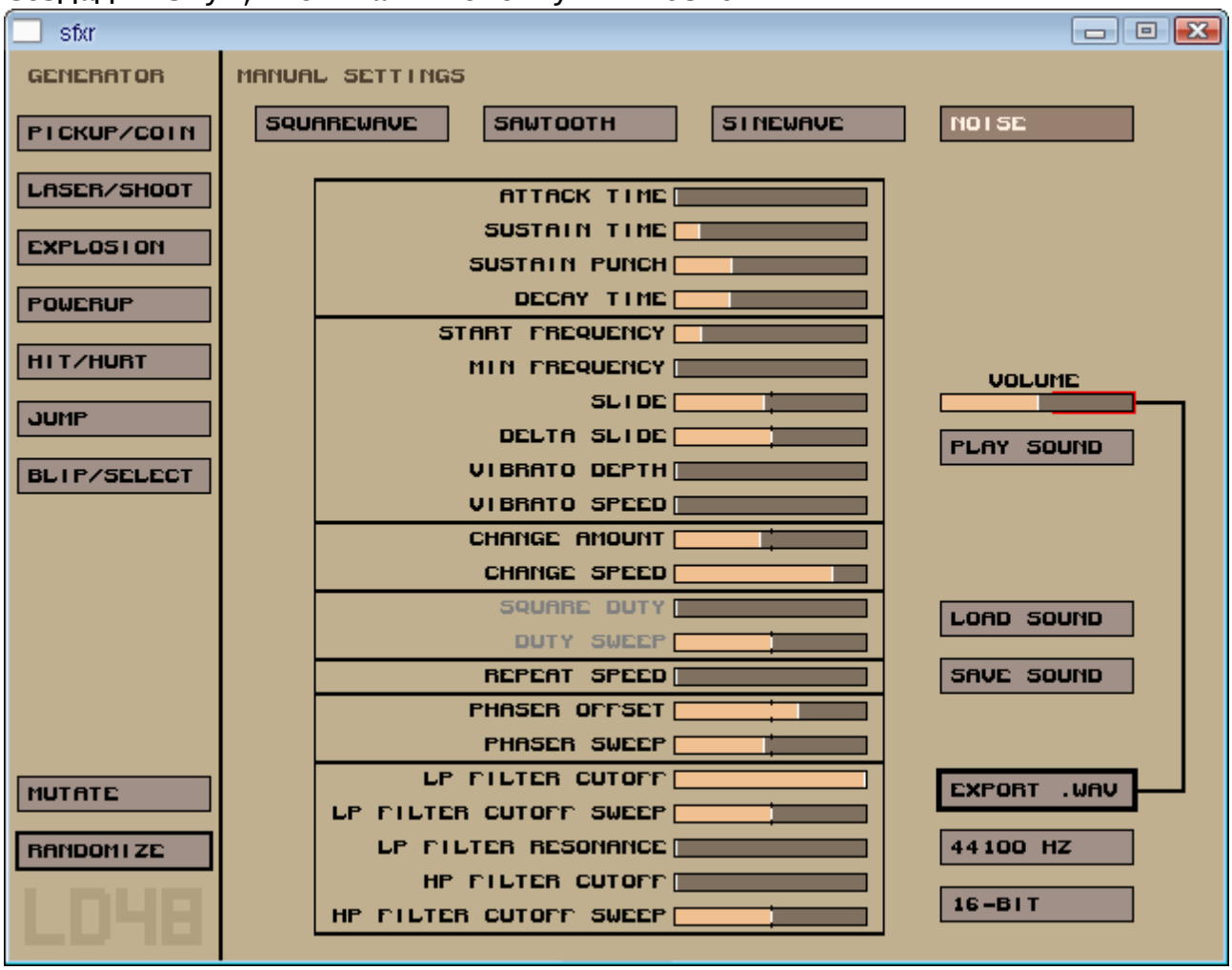
Теперь будем использовать эти обновлённые x и y чтобы определить, в какой же шарик мы попали. Этот шарик будем смещать сильно вниз и менять горизонтальную позицию на случайную.

```
FUNC TOUCHDOWN(INT x, INT y, INT ptr, INT btn) RETURNS BOOL
  x -= GETW()/2 + 24
  y = -y
  y += GETH()/2 - 212
  INT S = 48
  FOR (INT I=0; I<BALL.LENGTH; I++)
    IF (x<BALL[I].GETX()+S AND x>BALL[I].GETX()-S) THEN
      IF (y<BALL[I].GETY()+S AND y>BALL[I].GETY()-S) THEN
        DEBUG("BOOM")
        INT NEWX = RND(GETW()/64)*64+64 - GETW()/2
        INT NEWY = Math.round(BALL[I].GETY()) - GETH()
        BALL[I].SETPOS(NEWX, NEWY)
      ENDIF
    ENDIF
  NEXT
  RETURN TRUE
ENDFUNC
```

Полный листинг обновлённой программы:

```
SHEET[][] SHBALL
SPRITE[] BALL
FLOAT ANIMCOUNTER = 0, ANIMSPEED = 30

FUNC INIT()
  SHBALL = IMAGE_SPLIT(IMAGE_LOAD("balloon.png"), 32, 128)
  BALL = NEW SPRITE[6]
  INT OFFSET = 0
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I] = NEW SPRITE(SHBALL[0][0])
    INT NEWX = RND(GETW()/64)*64+64 -GETW()/2
    INT NEWY = -GETH()/2 - OFFSET
    BALL[I].SETPOS(NEWX, NEWY-GETH())
    BALL[I].SCALE(2)
    OFFSET += 64
  NEXT
ENDFUNC

FUNC CYCLE()
  RENDER()
  MOVEBALLS()
  ANIMATE()
ENDFUNC

FUNC RENDER()
  CLS(SKY)
  BEGIN_SPRITES()
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I].DRAW()
  NEXT
  END_SPRITES()
ENDFUNC

FUNC MOVEBALLS()
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I].MOVE(0,GETDELTA()*200)
  NEXT
ENDFUNC

FUNC ANIMATE()
  ANIMCOUNTER += GETDELTA()*100
  IF (ANIMCOUNTER > ANIMSPEED) THEN
    ANIMCOUNTER = 0
  ELSE
    RETURN
  ENDIF
  FOR (INT I=0; I<BALL.LENGTH; I++)
    INT RF = RND(2)
    BALL[I].ASSIGN(SHBALL[RF][0])
  NEXT
ENDFUNC

FUNC TOUCHDOWN(INT x, INT y, INT ptr, INT btn) RETURNS BOOL
  x -= GETW()/2 + 24
  y = -y
  y += GETH()/2 - 212
  INT S = 48
  FOR (INT I=0; I<BALL.LENGTH; I++)
    IF (x<BALL[I].GETX()+S AND x>BALL[I].GETX()-S) THEN
      IF (y<BALL[I].GETY()+S AND y>BALL[I].GETY()-S) THEN
        DEBUG("BOOM")
        INT NEWX = RND(GETW()/64)*64-64 - GETW()/2
        INT NEWY = Math.round(BALL[I].GETY()) - GETH()
        BALL[I].SETPOS(NEWX, NEWY)
      ENDIF
    ENDIF
  NEXT
  RETURN TRUE
ENDFUNC
```

Заметьте строчку DEBUG(“BOOM”). На её место надо вставить проигрывание звука лопающегося шарика, а также показ анимации из двух последних кадров. Чем мы и займемся.
Создадим звук, потыкав кнопочку EXPLOSION:



Экспортируем как boom.wav в папку *data/assets/*
Вносим изменения в код:

```
SHEET[][] SHBALL
SPRITE[] BALL
FLOAT ANIMCOUNTER = 0, ANIMSPEED = 30
SOUND BOOM

FUNC INIT()
  BOOM = SOUND_LOAD("boom.wav")
```

Вместо DEBUG(“BOOM”) пишем:

```
FLOAT RPITCH = 0.8f + RND(40)/100f
BOOM.PLAY(0.5f,RPITCH,0)
```

Можно было бы написать просто BOOM.PLAY(), но здесь мы уменьшили громкость вполовину, и сделали случайное изменение высоты для каждого звука для разнообразия.
Теперь добавим лопающиеся шары. Для этого заведем отдельный массив спрайтов:

```
SPRITE[] BALL, XBALL

FUNC INIT()
  BOOM = SOUND_LOAD("boom.wav")
  SHBALL = IMAGE_SPLIT(IMAGE_LOAD("balloon.png"), 32, 128)
  BALL = NEW SPRITE[6]
  XBALL = NEW SPRITE[6]
  INT OFFSET = 0
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I] = NEW SPRITE(SHBALL[0][0])
    XBALL[I] = NEW SPRITE(SHBALL[2][0])
    // прячем за пределы экрана:
    XBALL[I].SETPOS(GETW(),0)
    // увеличиваем как и целые шарики
    XBALL[I].SCALE(2)
```

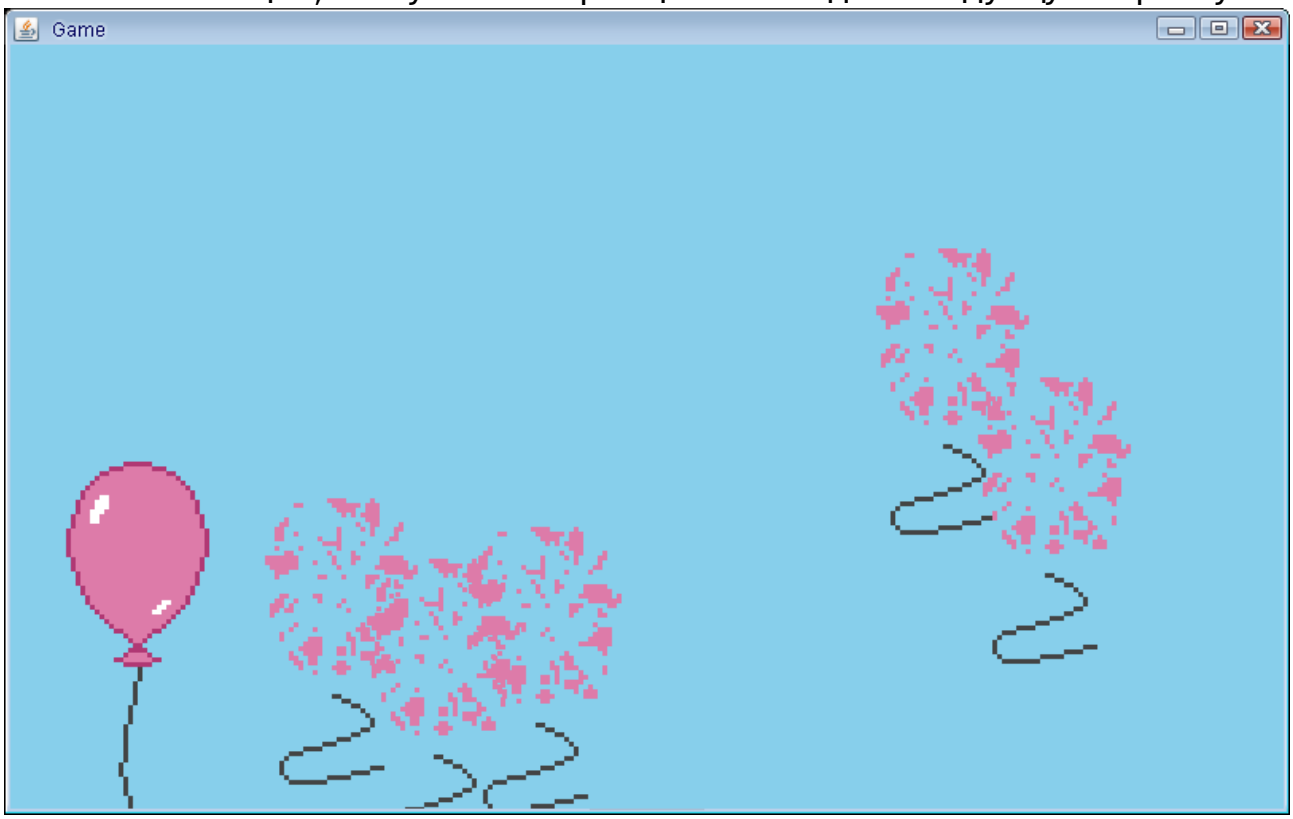
Рядом с проигрыванием звука пишем добавление лопнувшего шарика:

```
    BOOM.PLAY(0.5f,RPITCH,0)
    XBALL[I].SETPOS(BALL[I].GETX(),BALL[I].GETY())
```

И добавляем их в отрисовку:

```
FUNC RENDER()
  CLS(SKY)
  BEGIN_SPRITES()
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I].DRAW()
    XBALL[I].DRAW()
```

После компиляции, запуска и пары щелчков видим следующую картину:



Займемся анимацией лопнувших шаров и их удалением.

Для этого создадим отдельную функцию REMOVEBALLS.

В самом верху:

```
INT MAXBALLS = 6
FLOAT[] XSTATE = NEW FLOAT[MAXBALLS]
```

Этот массив будет хранить состояние каждого лопнувшего шара.
-1: лопнувший шар не активен, не виден на экране
>0: шар активен, состояние увеличивается на GETDELTA()*1500
0-100: активен, показывать первый кадр
100-200: активен, показывать второй кадр
>200: вернуть первый кадр, убрать шар за пределы экрана и вернуть состояние в неактивное (-1)
Вот функция REMOVEBALLS:

```
FUNC REMOVEBALLS()
  FOR (INT I=0; I<BALL.LENGTH; I++)
    IF (XSTATE[I]<0) CONTINUE
    XSTATE[I] += GETDELTA()*1500
    IF (XSTATE[I] > 100 AND XSTATE[I]<200) THEN
      XBALL[I].ASSIGN(SHBALL[3][0])
    ENDIF
    IF (XSTATE[I] > 200) THEN
      XBALL[I].ASSIGN(SHBALL[2][0])
      XSTATE[I] = -1
      XBALL[I].SETPOS(GETW(),0)
    ENDIF
  NEXT
ENDFUNC
```

Изменения в функции TOUCHDOWN (лопнувший близнец шара стаёт активным):

```
BOOM.PLAY(0.5f,RPITCH,0)
XBALL[I].SETPOS(BALL[I].GETX(),BALL[I].GETY())
XSTATE[I] = 1
```

Хвост функции INIT (изначально все лопнувшие шары неактивны):

```
          XSTATE[I]=-1
          OFFSET += 64
        NEXT
      ENDFUNC
```

Введем счетчик лопнутых шаров. Вверху программы:

```
INT KILLS = 0
```

После BOOM.PLAY(0.5f,RPITCH,0):

```
KILLS += 1
```

В хвост функции RENDER:

```
  PRINT("SCORE: " + TXT(KILLS),-GETW()/2+15,GETH()/2)
  END_SPRITES()
ENDFUNC
```

Пора ввести проверку на соблюдение условия проигрыша:
Если шарик ускользнул вверх, то мы проиграли, обнуляем всё.

```
FUNC MOVEBALLS()
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I].MOVE(0,GETDELTA()*200)
    IF (BALL[I].GETY()>GETH()/2-150) THEN
      RESETGAME()
    ENDIF
  NEXT
ENDFUNC
```

Новая функция. По большей части скопирована из INIT:

```
FUNC RESETGAME()
  KILLS = 0
  INT OFFSET = 0
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I].ASSIGN(SHBALL[0][0])
    XBALL[I].ASSIGN(SHBALL[2][0])
    INT NEWX = RND(GETW()/64)*64+64 -GETW()/2
    INT NEWY = -GETH()/2 - OFFSET
    BALL[I].SETPOS(NEWX, NEWY-GETH())
    XBALL[I].SETPOS(GETW(),0)
    XSTATE[I]=-1
    OFFSET += OFFY
  NEXT
ENDFUNC
```

Изменения вверху:

```
INT MAXBALLS = 6, OFFY = 256
```

Пусть игра усложняется со временем (чем больше шаров игрок уже лопнул, тем быстрее летят новые шары):

```
FUNC MOVEBALLS()
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I].MOVE(0,GETDELTA()*(200+KILLS/3))
```

Запустим игру на Android: *make-all.bat*, ждём, затем *android-install.bat* (на подключенном устройстве должен быть включен режим отладки: Настройки –> Приложения -> Разработка). В списке приложений появится наш BaloonShooter (с одной l, опечатка). Играть довольно тяжело. Заменим в функции TOUCHDOWN RETURN на FALSE и еще кое-что, чтобы лучше разделить шары по высоте:

```
            INT NEWY = -GETH()*2
            NEWY = Math.round(NEWY/128)*128
            BALL[I].SETPOS(NEWX, NEWY)
          ENDIF
        ENDIF
      NEXT
      RETURN FALSE
    ENDFUNC
```

После этого можно щелкать шары несколькими пальцами одновременно.

Всё равно тяжело. Переименовываем функцию TOUCHDOWN в TOUCHDRAGGED:

```
FUNC TOUCHDRAGGED(INT x, INT y, INT ptr) RETURNS BOOL
```

Теперь шары можно рассекать пальцами/зажатой мышью.

Для разнообразия сделаем шары цветными.
Вверху:

```
COLOR[] BCOLS = NEW COLOR[MAXBALLS]
```

В INIT:

```
        OFFSET += OFFY
        BCOLS[I] = RANDCOLOR()
        BALL[I].TINT(BCOLS[I])
        XBALL[I].TINT(BCOLS[I])
      NEXT
    ENDFUNC
```

То же самое в RESETGAME:

```
        OFFSET += OFFY
        BCOLS[I] = RANDCOLOR()
        BALL[I].TINT(BCOLS[I])
        XBALL[I].TINT(BCOLS[I])
      NEXT
    ENDFUNC
```

В REMOVEBALLS (чтобы в следующий раз лопнул новым цветом):

```
     XBALL[I].SETPOS(GETW(),0)
     XBALL[I].TINT(BCOLS[I])
```

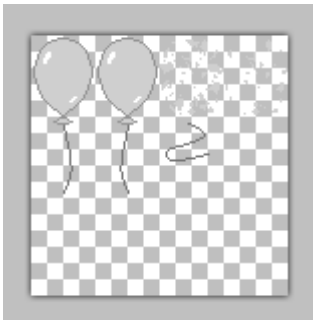
В TOUCHDRAGGED (готовим новый шар):

```
       BALL[I].SETPOS(NEWX, NEWY)
       BCOLS[I] = RANDCOLOR()
       BALL[I].TINT(BCOLS[I])
```

Изображение data/assets/balloon.png нужно сделать чернобелым и сделать поярче:

Функция RANDCOLOR:

```
FUNC RANDCOLOR() RETURNS COLOR
  FLOAT R = RND(2)/2f + 0.5f
  FLOAT G = RND(2)/2f + 0.5f
  FLOAT B = RND(2)/2f + 0.5f
  COLOR c = NEW COLOR(R,G,B,1)
  RETURN c
ENDFUNC
```



Сделаем учет рекордов:

```
INT HIGHSCORE = 0

FUNC INIT()
  HIGHSCORE = LOAD_INT("BS_HIGHSCORE",0)
  ...

FUNC RESETGAME()
  IF (KILLS > HIGHSCORE) THEN
    HIGHSCORE = KILLS
  ENDIF
  SAVE_INT("BS_HIGHSCORE",HIGHSCORE)
  SAVE_PREFS()
  KILLS = 0
  ...

FUNC RENDER()
 ...
 PRINT("SCORE: " + TXT(KILLS),-GETW()/2+15,GETH()/2)
 PRINT("HIGHSCORE: "+TXT(HIGHSCORE),-GETW()/2+15,GETH()/2-20)
```

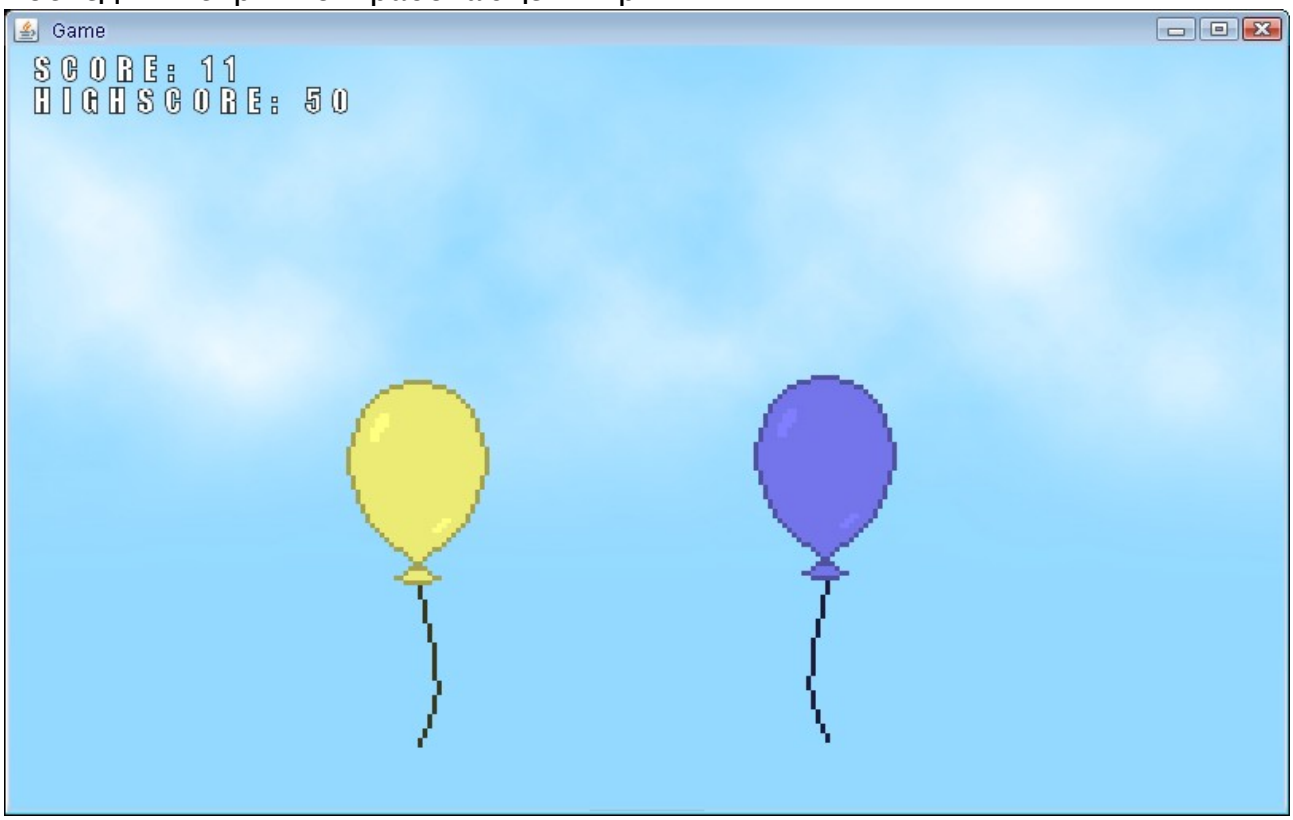
TODO
Много еще чего можно внести в этот проект, многое исправить, например:

- шары всё еще прячутся друг за друга, так что можно сделать условную сетку под экраном и заполнять/проверять перед заполнением ячейки сетки, расставляя новые шары.
- избежать появления скучного серого шара, подправив функцию генерации нового цвета
- перерисовать спрайт-лист в большем разрешении, чтобы не увеличивать вдвое
- движущиеся облака на фоне
- показывать сообщение Game over или типа того
- стартовое меню, логотип, фоновая музыка

Облака пожалуй, добавлю, но статические:

```
IMAGE clouds

FUNC INIT()
  clouds = IMAGE_LOAD("sky.png")
  ...

FUNC RENDER()
  CLS(SKY)
  BEGIN_SPRITES()
  DRAW(clouds,-GETW()/2,-GETH()/2,GETW(),GETH(),0,1,1,0)
```

Последний скриншот работающей игры:



Дальше улучшать уже не буду, лучше покажу как разбить код на отдельные файлы. Вот весь код, разрезанный на файлы:

a1.txt:

```
SHEET[][] SHBALL
SPRITE[] BALL, XBALL
FLOAT ANIMCOUNTER = 0, ANIMSPEED = 30
INT MAXBALLS = 6, OFFY = 256
FLOAT[] XSTATE = NEW FLOAT[MAXBALLS]
COLOR[] BCOLS = NEW COLOR[MAXBALLS]
SOUND BOOM
INT KILLS = 0
INT HIGHSCORE = 0
IMAGE clouds

FUNC INIT()
  INIT_GAME()
ENDFUNC

FUNC CYCLE()
  RENDER()
  MOVEBALLS()
  ANIMATE()
  REMOVEBALLS()
  CAPFPS(120)
ENDFUNC
```

init.txt:

```
FUNC INIT_GAME()
  clouds = IMAGE_LOAD("sky.png")
  HIGHSCORE = LOAD_INT("BS_HIGHSCORE",0)
  BOOM = SOUND_LOAD("boom.wav")
  SHBALL = IMAGE_SPLIT(IMAGE_LOAD("balloon.png"), 32, 128)
  BALL = NEW SPRITE[MAXBALLS]
  XBALL = NEW SPRITE[MAXBALLS]
  INT OFFSET = 0
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I] = NEW SPRITE(SHBALL[0][0])
    XBALL[I] = NEW SPRITE(SHBALL[2][0])
    INT NEWX = GETNEWBALLX()
    INT NEWY = -GETH()/2 - OFFSET
    BALL[I].SETPOS(NEWX, NEWY-GETH())
    XBALL[I].SETPOS(GETW(),0)
    BALL[I].SCALE(2)
    XBALL[I].SCALE(2)
    XSTATE[I]=-1
    OFFSET += OFFY
    BCOLS[I] = RANDCOLOR()
    BALL[I].TINT(BCOLS[I])
    XBALL[I].TINT(BCOLS[I])
  NEXT
ENDFUNC
```

reset.txt:

```
FUNC RESETGAME()
  IF (KILLS > HIGHSCORE) THEN
    HIGHSCORE = KILLS
  ENDIF
  SAVE_INT("BS_HIGHSCORE",HIGHSCORE)
  SAVE_PREFS()
  KILLS = 0
  INT OFFSET = 0
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I].ASSIGN(SHBALL[0][0])
    XBALL[I].ASSIGN(SHBALL[2][0])
    INT NEWX = GETNEWBALLX()
    INT NEWY = -GETH()/2 - OFFSET
    BALL[I].SETPOS(NEWX, NEWY-GETH())
    XBALL[I].SETPOS(GETW(),0)
    XSTATE[I]=-1
    OFFSET += OFFY
    BCOLS[I] = RANDCOLOR()
    BALL[I].TINT(BCOLS[I])
    XBALL[I].TINT(BCOLS[I])
  NEXT
ENDFUNC
```

render.txt:

```
FUNC RENDER()
  CLS(SKY)
  BEGIN_SPRITES()
  DRAW(clouds,-GETW()/2,-GETH()/2,GETW(),GETH(),0,1,1,0)
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I].DRAW()
    XBALL[I].DRAW()
  NEXT
  PRINT("SCORE: " + TXT(KILLS),-GETW()/2+15,GETH()/2)
  PRINT("HIGHSCORE: "+TXT(HIGHSCORE),-GETW()/2+15,GETH()/2-20)
  END_SPRITES()
ENDFUNC
```

anim.txt:

```
FUNC ANIMATE()
  ANIMCOUNTER += GETDELTA()*100
  IF (ANIMCOUNTER > ANIMSPEED) THEN
    ANIMCOUNTER = 0
  ELSE
    RETURN
  ENDIF
  FOR (INT I=0; I<BALL.LENGTH; I++)
    INT RF = RND(2)
    BALL[I].ASSIGN(SHBALL[RF][0])
  NEXT
ENDFUNC
```

balls.txt:

```
FUNC MOVEBALLS()
  FOR (INT I=0; I<BALL.LENGTH; I++)
    BALL[I].MOVE(0,GETDELTA()*(200+KILLS/2))
    IF (BALL[I].GETY()>GETH()/2-150) THEN
      RESETGAME()
    ENDIF
  NEXT
ENDFUNC

FUNC REMOVEBALLS()
  FOR (INT I=0; I<BALL.LENGTH; I++)
    IF (XSTATE[I]<0) CONTINUE
    XSTATE[I] += GETDELTA()*1500
    IF (XSTATE[I] > 100 AND XSTATE[I]<200) THEN
      XBALL[I].ASSIGN(SHBALL[3][0])
    ENDIF
    IF (XSTATE[I] > 200) THEN
      XBALL[I].ASSIGN(SHBALL[2][0])
      XSTATE[I] = -1
      XBALL[I].SETPOS(GETW(),0)
      XBALL[I].TINT(BCOLS[I])
    ENDIF
  NEXT
ENDFUNC
```

misc.txt:

```
FUNC GETNEWBALLX() RETURNS INT
  RETURN RND(GETW()/64)*64 + 48 - GETW()/2
ENDFUNC

FUNC RANDCOLOR() RETURNS COLOR
  FLOAT R = RND(2)/2f + 0.5f
  FLOAT G = RND(2)/2f + 0.5f
  FLOAT B = RND(2)/2f + 0.5f
  COLOR c = NEW COLOR(R,G,B,1)
  RETURN c
ENDFUNC
```

input.txt:

```
FUNC TOUCHDRAGGED(INT x, INT y, INT ptr) RETURNS BOOL
  x -= GETW()/2 + 24
  y = -y
  y += GETH()/2 - 212
  INT S = 48
  FOR (INT I=0; I<BALL.LENGTH; I++)
    IF (x<BALL[I].GETX()+S AND x>BALL[I].GETX()-S) THEN
```

```
      IF (y<BALL[I].GETY()+S AND y>BALL[I].GETY()-S) THEN
        FLOAT RPITCH = 0.8f + RND(40)/100f
        BOOM.PLAY(0.5f,RPITCH,0)
        KILLS++
        XBALL[I].SETPOS(BALL[I].GETX(),BALL[I].GETY())
        XSTATE[I] = 1
        INT NEWX = GETNEWBALLX()
        INT NEWY = -GETH()*2
        NEWY = Math.round(NEWY/128)*128 - I*OFFY
        BALL[I].SETPOS(NEWX, NEWY)
        BCOLS[I] = RANDCOLOR()
        BALL[I].TINT(BCOLS[I])
      ENDIF
    ENDIF
  NEXT
  RETURN FALSE
ENDFUNC
```

Все файлы должны содержать пустую строчку в конце или вначале (чтобы хорошо склеились). В один файл можно совать сколько угодно функций. Объявления глобальных переменных, функции INIT и CYCLE должны оставаться в a1.txt.

Теперь копипаста из README:

Чтобы модульная структура работала, вы должны

1) Положить в папку source (рядом со всеми исходниками) файл combine.bat:

```
@ECHO OFF
del %1\main.txt
type %1\*.txt >> %1\all
move %1\all %1\main.txt
```

Запускать его вручную нет смысла.

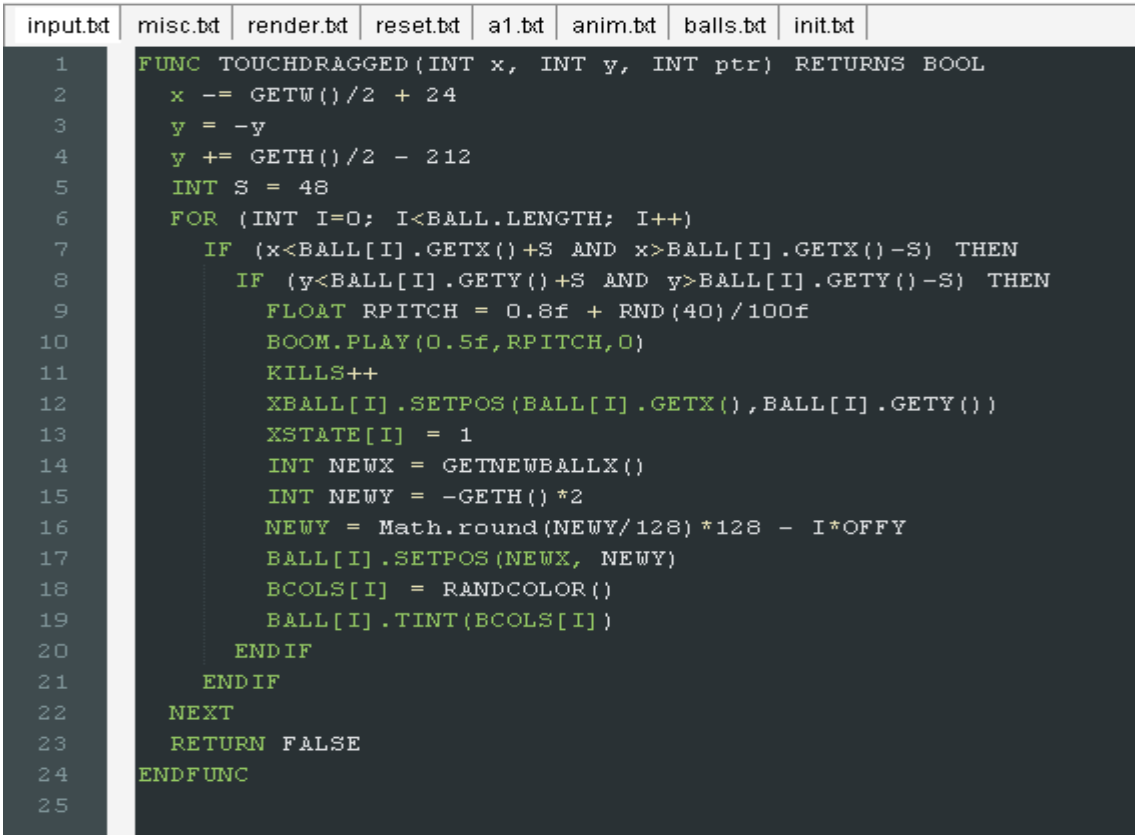
2) В make-*.bat файлы, после строчки
Set Appname=YourAppName
добавить две строчки (если их там еще нет, если есть, то пусть остаются всегда):

```
CALL %PROJECT%\source\combine.bat %PROJECT%\source >%PROJECT%\error.log
CLS
```

Вот это всё, что надо сделать для того, чтобы использовать модульную структуру. Если вы захотите вернуться к обычной монолитной структуре (всё в main.txt), удалите из папки source все файлы, включая combine.bat, кроме main.txt. Из make-*.bat те две строчки убирать не нужно, они не повредят.

Т.е. по-прежнему запускаем make-desktop или make-all, но функции у нас теперь в разных файлах.

Теперь в нашем проекте нет файлов длиннее 25 строчек и можно пользоваться удобным текстовым редактором со вкладками.




```
FUNC TOUCHDRAGGED(INT x, INT y, INT ptr) RETURNS BOOL
  x -= GETW()/2 + 24
  y = -y
  y += GETH()/2 - 212
  INT S = 48
  FOR (INT I=0; I<BALL.LENGTH; I++)
    IF (x<BALL[I].GETX()+S AND x>BALL[I].GETX()-S) THEN
      IF (y<BALL[I].GETY()+S AND y>BALL[I].GETY()-S) THEN
        FLOAT RPITCH = 0.8f + RND(40)/100f
        BOOM.PLAY(0.5f,RPITCH,0)
        KILLS++
        XBALL[I].SETPOS(BALL[I].GETX(),BALL[I].GETY())
        XSTATE[I] = 1
        INT NEWX = GETNEWBALLX()
        INT NEWY = -GETH()*2
        NEWY = Math.round(NEWY/128)*128 - I*OFFY
        BALL[I].SETPOS(NEWX, NEWY)
        BCOLS[I] = RANDCOLOR()
        BALL[I].TINT(BCOLS[I])
      ENDIF
    ENDIF
  NEXT
  RETURN FALSE
ENDFUNC

```

Этот проект вы найдете в *C:\jdummy2d\tutorials\BaloonShooter.zip*