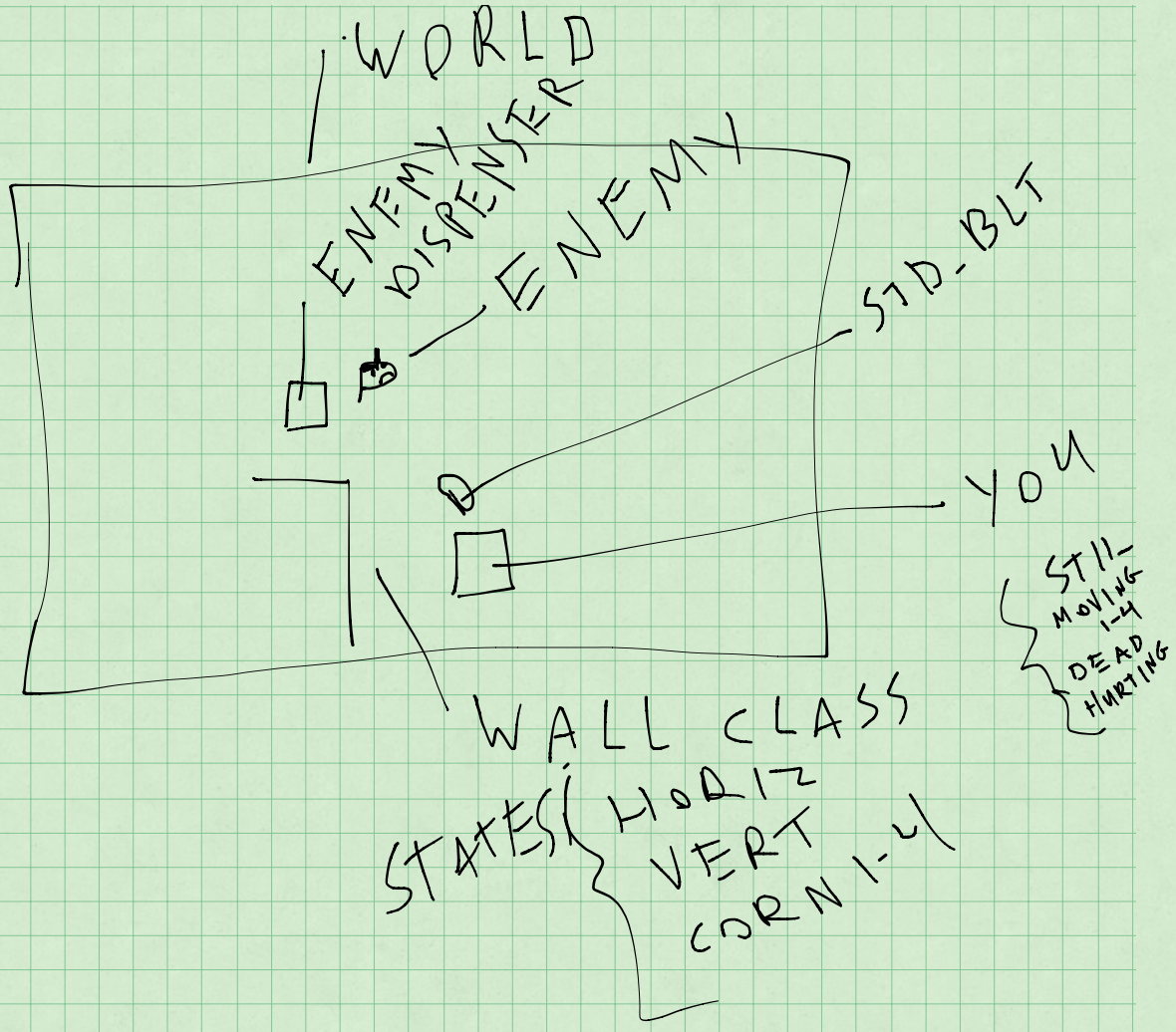
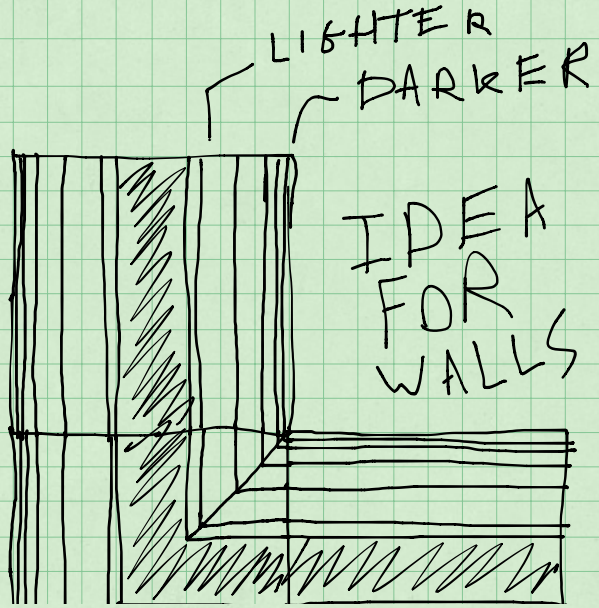
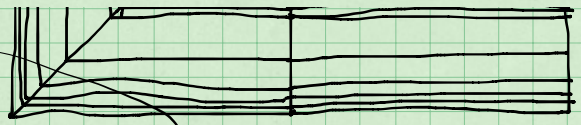



diferent textures cause different actions to the
hhhhhhhhhhhhhhh



## WORLD

LISTS
- WALLS
- FLOORS
- ENEMY DISPENSERS
- BULLETS
- ENEMY BULLETS

HERO

MAKE CONSTRUCTOR FILL ARRAY

## WORLD RENDERER

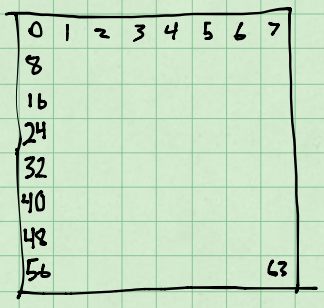
WORLD TEXTURE REGION

CHAR TEXTUR
CHAR REG

## LEVEL

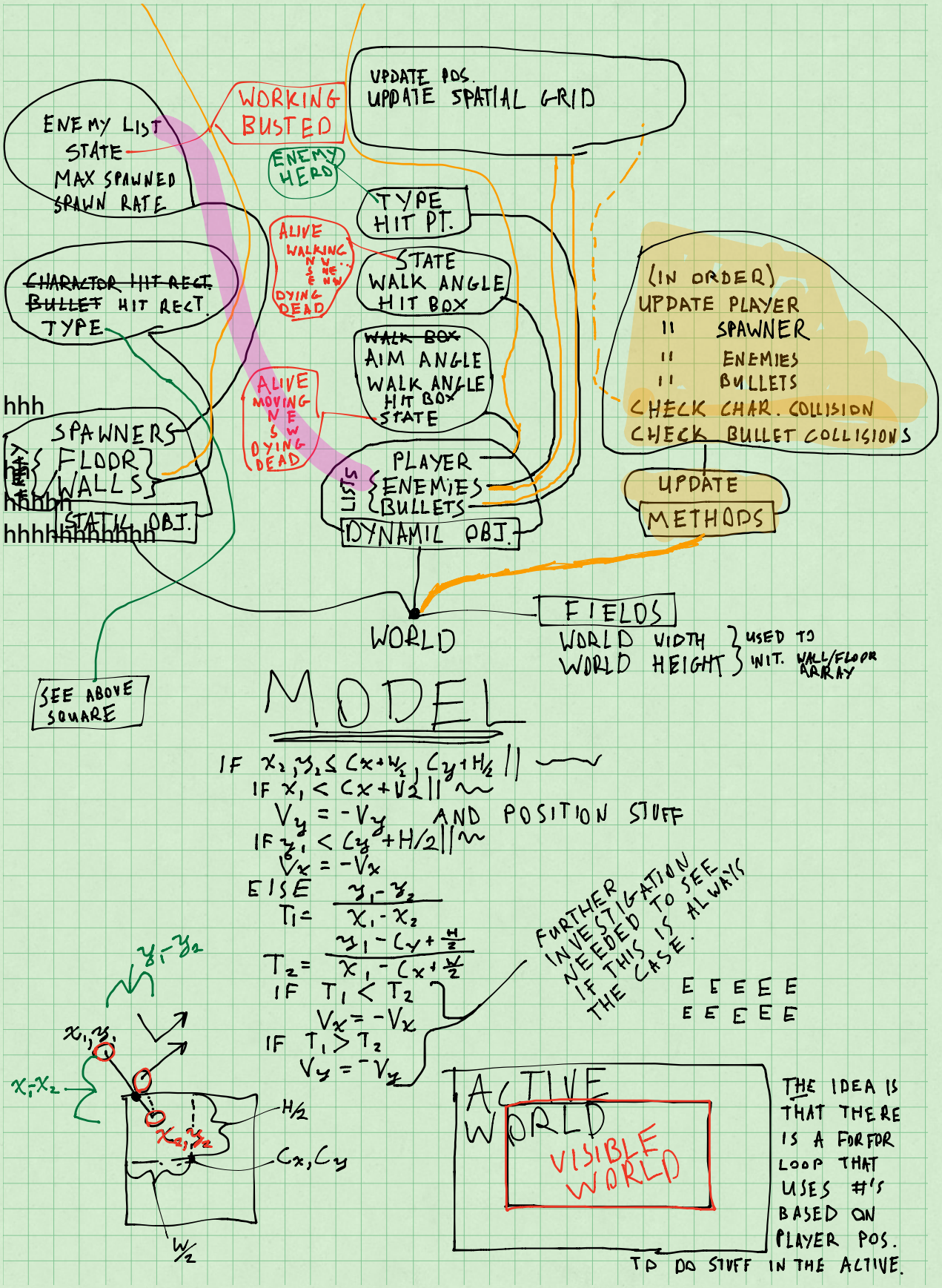
INITIAL LISTS
- WALLS
- FLOORS
- ENEMY DISPENSERS

TEXTURE REGIONS




PUBLIC INT TYPE

UPDATE POS.
DETEMINE UPDATE RANGE (1.5 x SCR.)


ENEMY LIST
STATE
MAX SPAWNED
SPAWN RATE
WORKING
BUSTED
UPDATE POS.
UPDATE SPATIAL GRID
ENEMY
HERO
TYPE
HIT PT.
ALIVE
WALKING
N
NW
W
SW
S
SE
E
NE
DYING
DEAD
STATE
WALK ANGLE
HIT BOX
~~CHARACTOR HIT RECT.~~
~~BULLET~~ HIT RECT.
TYPE
~~WALK BOX~~
AIM ANGLE
WALK ANGLE
HIT BOX
STATE
(IN ORDER)
UPDATE PLAYER
" SPAWNER
" ENEMIES
" BULLETS
CHECK CHAR. COLLISION
CHECK BULLET COLLISIONS
ALIVE
MOVING
N
E
S
W
DYING
DEAD
hhh
SPAWNERS
LISTS
FLOOR
WALLS
STATIC OBJ.
hhhhhhhhhhhh
PLAYER
LISTS
ENEMIES
BULLETS
DYNAMIC OBJ.
UPDATE
METHODS
WORLD
FIELDS
WORLD WIDTH
WORLD HEIGHT
USED TO INIT. WALL/FLOOR ARRAY
SEE ABOVE SQUARE
MODEL
IF $x_2, y_2 \leq C_x + W/2, C_y + H/2$ ||
IF $x_1 < C_x + V_2$ ||
$V_y = -V_y$ AND POSITION STUFF
IF $y_1 < C_y + H/2$ ||
$V_x = -V_x$
ELSE $T_1 = \frac{y_1 - y_2}{x_1 - x_2}$
$T_2 = \frac{y_1 - C_y + \frac{H}{2}}{x_1 - C_x + \frac{W}{2}}$
IF $T_1 < T_2$
$V_x = -V_x$
IF $T_1 > T_2$
$V_y = -V_y$
$y_1 - y_2$
$x_1, y_1$
$x_1 - x_2$
$x_2, y_2$
$H/2$
$C_x, C_y$
$W/2$
FURTHER INVESTIGATION NEEDED TO SEE IF THIS IS ALWAYS THE CASE.
E E E E E
E E E E E
ACTIVE
WORLD
VISIBLE
WORLD
THE IDEA IS THAT THERE IS A FOR FOR LOOP THAT USES #'s BASED ON PLAYER POS.
TO DO STUFF IN THE ACTIVE.

# WORLD

## FIELDS

```
INT                          WORLD HEIGHT/WIDTH
FLOOR/WALLS[ ][ ]            MAP
LIST<SPAWNERS>               SPAWNERS
LIST<BULLETS>                BULLETS      } ASSEMBLED IN CONSTRUCTOR
HERO                         HERO
SPATIAL HASHGRID             GRID  — ASSEMBLE STATIC IN CONSTRUCTOR
```

TO BE MODIFIED TO INCLUDE DIFFERENT TYPES OF LISTS

## METHODS

```
UPDATE(dt)
   INT x1 = (INT)(HERO.Pos.x - 6.0f)
       x2 =        "          + 6.0f
       y1 =        "      .y  - 3.5f
       y2 =        "          + 3.5f
```

OR WHATEVER NUMBERS ARE DECIDED ON.

```
   HERO.UPDATE(dt)
   FOR(SPAWNERS)
      SPAWNER.UPDATE(x1,x2,y1,y2,dt)
      FOR(SPAWNER.SPAWN)
         SPAWN.UPDATE(x1,y1,x2,y2,dt)
         GRID.ADDDYNAMIC(SPAWN)
         LIST<OBJECT> COLLIDERS = GRID.GETPOTENTIALSTATICCOLLIDERS(SPAWN)
         FOR(COLLIDERS)
            COLLIDER.CHECKGROUNDCOLLISION(SPAWN)

   FOR(BULLETS)
      BULLET.UPDATE(x1,x2,y1,y2,dt)
      LIST<OBJECT> COLLIDERS GRID.GETPOTENTIALCOLLIDERS(BULLET)
      FOR(COLLIDERS)
         IF(COLLIDER INSTANCEOF WALL)
            COLLIDER.CHECKBULLETCOLLISION(BULLET)
         ELSE
            COLLIDER.CHECKBULLETCOLLISION(BULLET)
            COLLIDER.HIT()
```

THESE GET UPDATED FIRST AND THEREFORE THE DYNAMIC LISTS ONLY HAVE SPAWNS.

---

# SPAWNER

## FIELDS

LIST<SPAWN>  SPAWNS  ← SPAWN POOL?
INT  STATE

INT  ALIVE = 0
INT  SPAWNING = 1
INT  HURT = 2
INT  DYING = 3
INT  DEAD = 4

INT  FRAME (TODO)

FLOAT  TICKTIMER
FLOAT  TICK_TIME

METHODS
UPDATE(x1, x2, y1, y2, dt)

IF (THIS.POS.x > x2 && < x1 ~~~)
  TICK TIMER += dt
  WHILE(TICKTIMER ≥ TICK_TIME)
    TICKTIMER -= TICK_TIME
    SPAWN  SPAWN = NEW SPAWN(POSITION OF SPAWNER)
    SPAWNS.ADD(SPAWN)

SPAWN  EXTENDS DYNAMIC GAME OBJECT  ← WILL BE MODIFIED TO INCLUDE CHECK BULLET COLLISIONS

FLOAT  WALKTICKTIME
FLOAT  WALKTICKER
INT  STATE

CONSTRUCTOR
POSITION
VELOCITY

INT  STOPPED
INT  WALKING
INT  SHOOTING?
INT  DYING
INT  DEAD

METHODS

UPDATE($x_1, x_2, y_1, y_2, dt$)

IF (THIS.POS.x > x2 && < x1 ~~~)
  $x_1 = x_2$
  $y_1 = y_2$
  $x_2 += V_x \cdot dt$
  $y_2 += V_y \cdot dt$
  CHECK COLLISIONS W/ WALLS()

CHECKCOLLISIONS W/WALLS()

# WALL/FLOOR EXTENDS GAME OBJECT

IS TO BE MODIFIED TO INCLUDE CHECKBULLETCOLLISIONS & CHECKGROUNDCOLLISIONS

## FIELDS

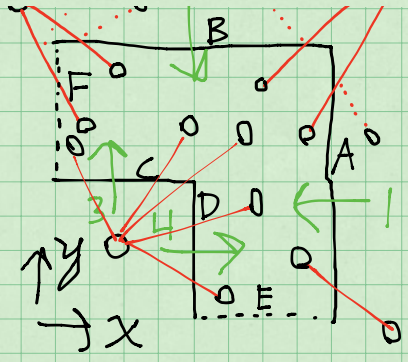
| | |
|---|---|
| INT | HORZ |
| INT | VERT |
| " | T_R |
| ⋮ | T_D |
| | T_L |
| | T_U |
| | E_R_U |
| | E_D_R |
| | E_L_D |
| | E_U_L |
| | S_R |
| | S_D |
| | S_L |
| | S_U |
| | CROSS |
| | FLOOR |

$(x<A \wedge y<B) \wedge ((x>D \wedge y>E) \| (x>F \wedge y>C))$

REFLECT ⇓

```
IF  Vx>0 || Vy>0
    CHECK  3 & 4
IF  Vx>0 || Vy<0
    CHECK  2 & 4
IF  Vx<0 || Vy>0
    CHECK  3 & 1
IF  Vx<0 || Vy<0
    CHECK  2 & 1
```

```
IF x<A && x>D && y<B && y>E
||
   Δx = x1 − x2
   Δy = y1 − y2
   IF Δx<0 && Δy<0
      IF x>D && y>C
         Vx = -Vx, Vy = -Vy
         x=D , y=C }
         RETURN
   IF x<D
```




$y_R$: $\frac{\Delta y}{\Delta x} < \frac{y_1 - B}{x_1 - A}$

$\Rightarrow \Delta y(x_1 - A) < \Delta x(y_1 - B)$

$x_R$: $\frac{\Delta y}{\Delta x} > \frac{y_1 - B}{x_1 - A}$

$\Delta y(x_1 - A) > \Delta x(y_1 - B)$

```
    Vy = -Vy
    y = C
ELSE
    Vx = -Vx
    x = D
RETURN
IF Δx<0 && Δy>0
  IF y > C
    Vy = -Vy
    y = B
  ELSE
    Vx = -Vx
    x = D
  RETURN
IF Δx>0 && Δy>0
  IF x > D
    Vx = -Vx
    x = A
  ELSE
    Vy = -Vy
    y = C
  RETURN
IF Δx>0 && Δx>0
  IF x1 < A
    Vy = -Vy
    y = B
    RETURN
  IF y1 < B
    Vx = -Vx
    x = A
    RETURN
  IF  Δy(x1-A) > Δx(y1-B)
    Vx = -Vx
    x = A
    RETURN
  IF  Δy(x1-A) < Δx(y1-B)
    Vy = -Vy
    y = B
  ELSE
    Vy = -Vy, Vx = -Vx
    x = A, y = B
  RETURN
```