1) Create a bit structure with bricks in the path of the ball

bit number

b_4	b_3	b_2	
4	3	2	
	↗	1	b_1
		0	b_0

↗ ball position and direction

4	:	$Target_X$	=	X_{pos}	-	$Delta_X$
		$Target_Y$	=	Y_{pos}	+	$Delta_Y$
3	:	$Target_X$	=	X_{pos}		
		$Target_Y$	=	Y_{pos}	+	$Delta_Y$
2	:	$Target_X$	=	X_{pos}	+	$Delta_X$
		$Target_Y$	=	Y_{pos}	+	$Delta_Y$
1	:	$Target_X$	=	X_{pos}	+	$Delta_X$
		$Target_Y$	=	Y_{pos}		
0	:	$Target_X$	=	X_{pos}	+	$Delta_X$
		$Target_Y$	=	Y_{pos}	-	$Delta_Y$

* if DeltaX=DeltaY, bits are flipped

	7	6	5	4	3	2	1	0
Byte	0	0	0	b_4	b_3	b_2	b_1	b_0

b_i : 1 if there **IS** a brick
0 if there is **NOT** a brick

1) Look for bricks at all b_i positions
And create a bit-encoded variable
2) Flip bits if DeltaX=DeltaY

2) Analyze bit structure for selecting collision case

Case 0 "No hit"

X	0	0
	↗	0
		X

Case 1 "Convex Corner"

X	0	1
	↗	0
		X

Case 2 "Y Brick"

X	1	X
	↗	0
		0

Case 3 "X Brick"

0	0	X
	↗	1
		X

Case 4 "Concave Corner" "Default*"

X	1	X
	↗	1
		X

↗ : Current position and direction of the ball
X : Don't care if there is a brick or not
0 : There is **NOT** a brick
1 : There **IS** a brick
: New Ball Position
: Brick to destroy

TRUTH TABLE

	bit 7	6	5	4	3	2	1	0	Mask Value	
Case0	X	X	X	X	0	0	0	X	$0E	$00
Case1	X	X	X	X	0	1	0	X	$0E	$04
Case2	X	X	X	X	1	X	0	0	$0B	$08
Case3	X	X	X	0	0	X	1	X	$1A	$02
Case4	X	X	X	X	1	X	1	X	$0A	$0A

*May be default, without mask/check

1) Get the brick collision structure
2) Apply Mask (logic AND)
3) Check for Value
4) Modify a variable to hold the collision case

* This case is also used when Case 2 and Case 3 destination is occupied. Yo may use it as default

3) Switch Case for destroying bricks and modifying Ball Deltas

	Brick to Destroy at X	Brick to Destroy at Y	New Ball Deltas D_X	New Ball Deltas D_Y
Case 0	N/A	N/A	=	=
Case 1	$X+D_X$	$Y+D_Y$	neg	neg
Case 2	X	$Y+D_Y$	=	neg
Case 3	$X+D_X$	Y	neg	=
Case 4	X	$Y+D_Y$	neg	neg

*neg = two's complement

1) Get the collision case variable
2) Perform a "switch case"
3) Destroy corresponding bricks
4) Modify the Ball Deltas

UFV- Computer Engineering Degree
Computer Architecture and Organization
"Arkanoid-Clone" collision detection method
v1.4 Daniel León - November 2024