[EXTERNAL EXPORT] View Exposure Measurement

*This doc gives an overview of the algorithm dedicated to measuring the changes in View Exposure respectively to the requirements of MRAID 3 (**7.5 exposureChange p 55**).*

Look at the code of ViewExposureChecker for the details.

Member variables

- @property (nonatomic, nonnull, strong, readonly) UIView *testedView;

  Holds the view, for which the exposure is measured.

- @property (nonatomic, assign, readwrite) CGRect clippedRect;

  Holds the part of the tested view, which is still visible after all clipping applied by the chain of parent views.

  Coordinate system: testedView.bounds

- @property (nonatomic, nonnull, strong) NSMutableArray<NSValue *> *obstructions;

  Rectangles of all obstructing views, projected onto testedView and clipped by clippedRect.

  Coordinate system: testedView.bounds

Initialization

- - (instancetype)initWithView:(UIView *)view

  testedView – assigned from the view argument.

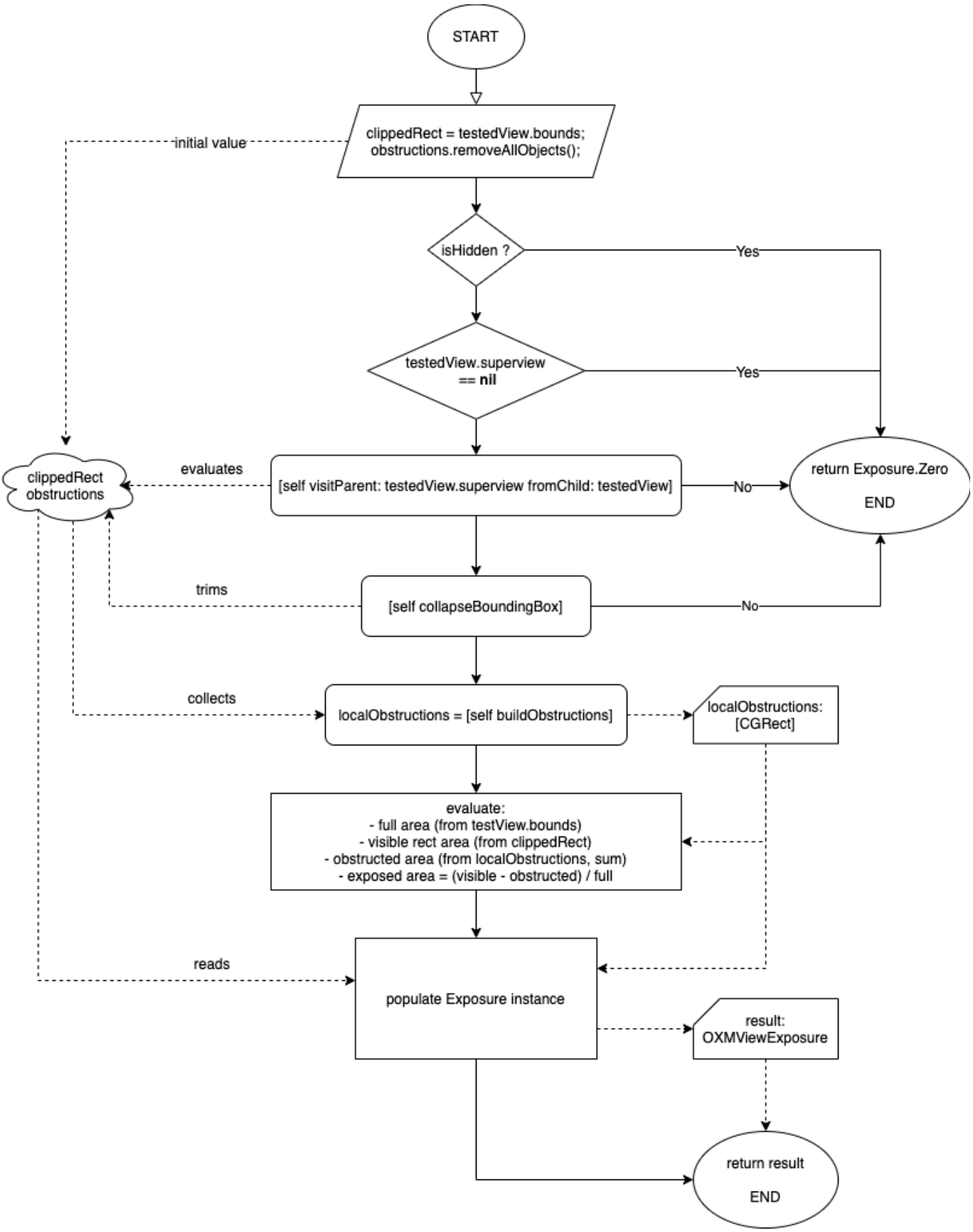
  obstructions – assigned a new empty mutable array.

  clippedRect – ignored.

Methods

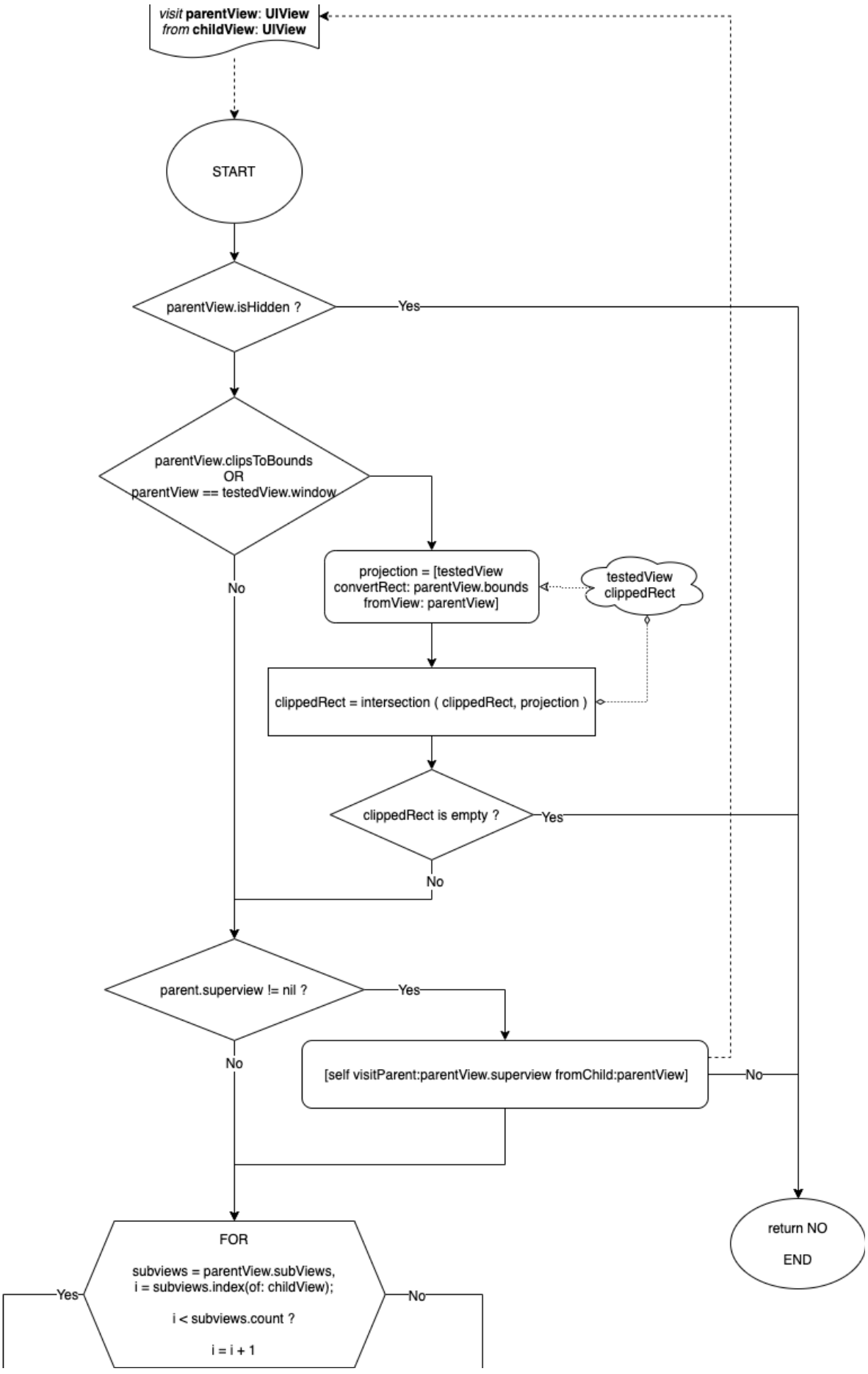
- (OXMViewExposure *)exposure

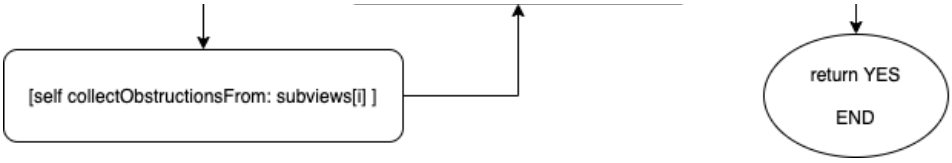
The primary method for evaluating the exposure.



- (BOOL)visitParent:(UIView *)parentView fromChild:(UIView *)childView

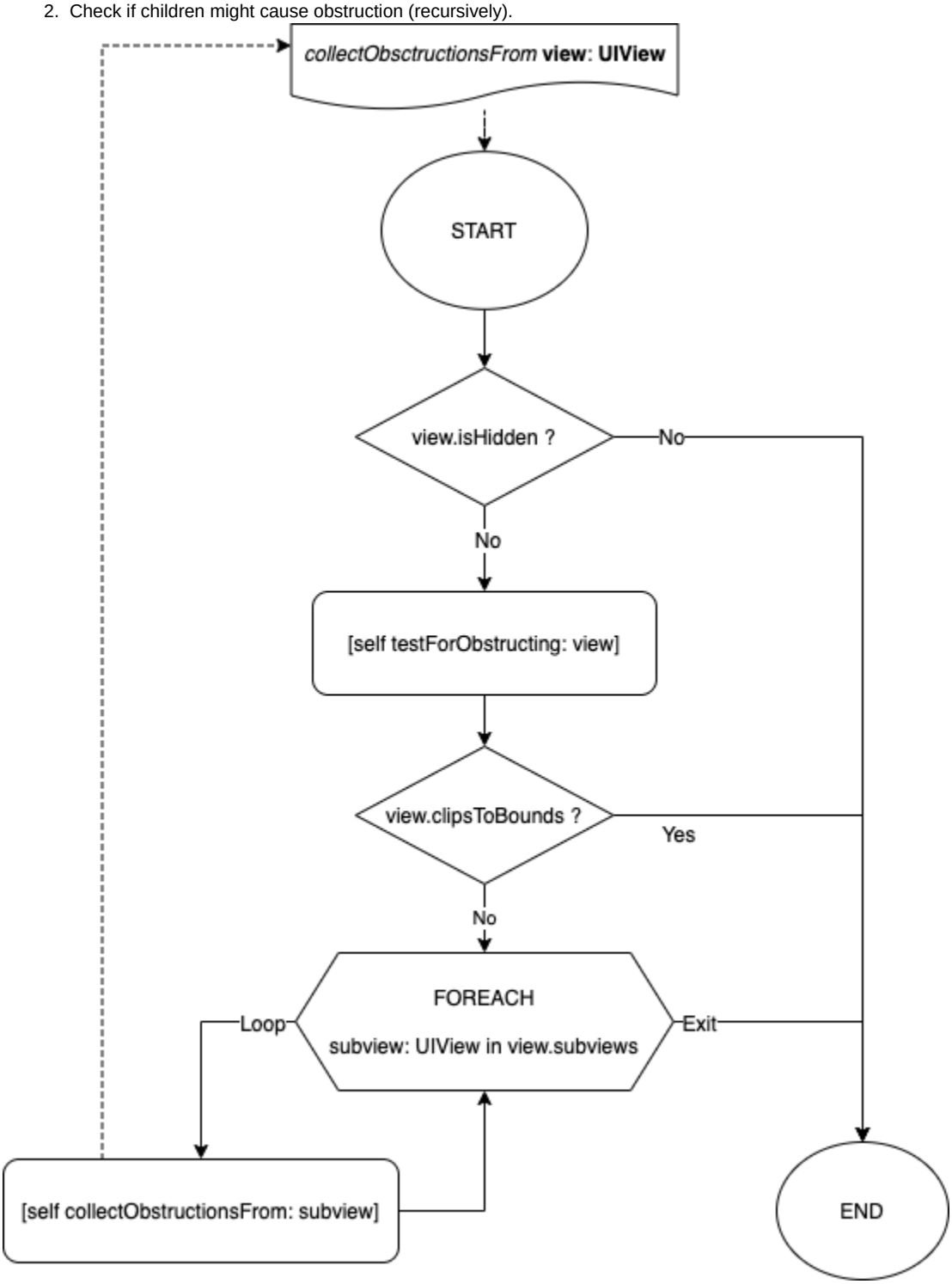
A recursive method called for each parent in the chain of superviews.


visit parentView: UIView
from childView: UIView
START
parentView.isHidden ?
Yes
parentView.clipsToBounds
OR
parentView == testedView.window
No
projection = [testedView
convertRect: parentView.bounds
fromView: parentView]
testedView
clippedRect
clippedRect = intersection (clippedRect, projection)
clippedRect is empty ?
Yes
No
parent.superview != nil ?
Yes
No
[self visitParent:parentView.superview fromChild:parentView]
No
return NO
END
FOR
subviews = parentView.subViews,
i = subviews.index(of: childView);
i < subviews.count ?
i = i + 1
Yes
No



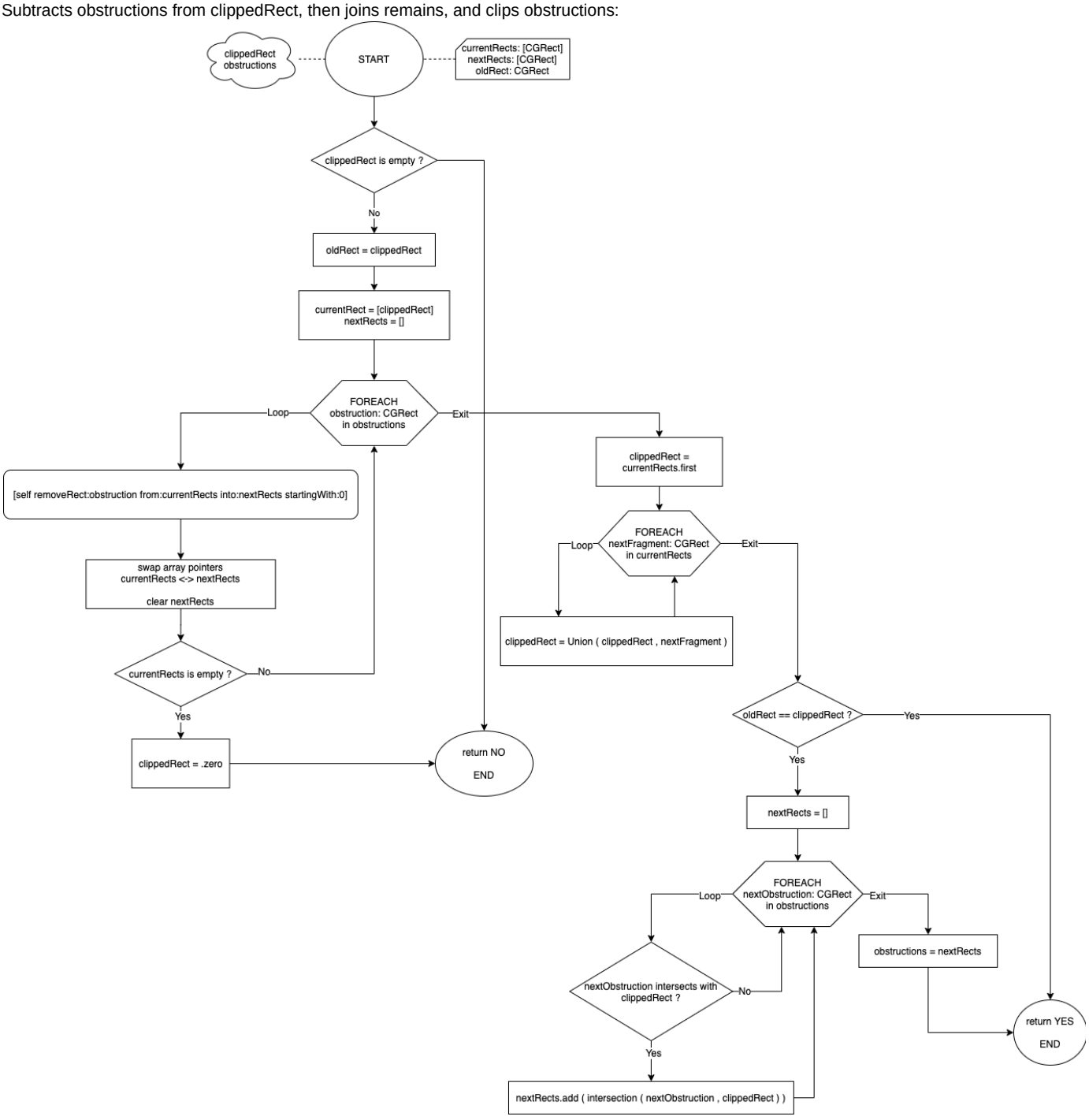
- (void)collectObstructionsFrom:(UIView *)view

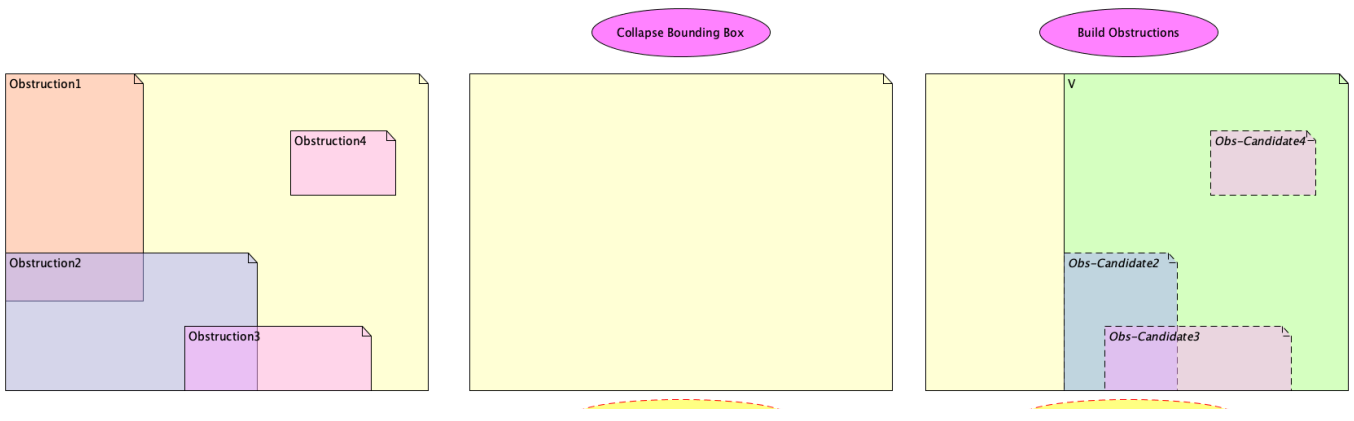
1. Check if view itself causes obstruction
2. Check if children might cause obstruction (recursively).

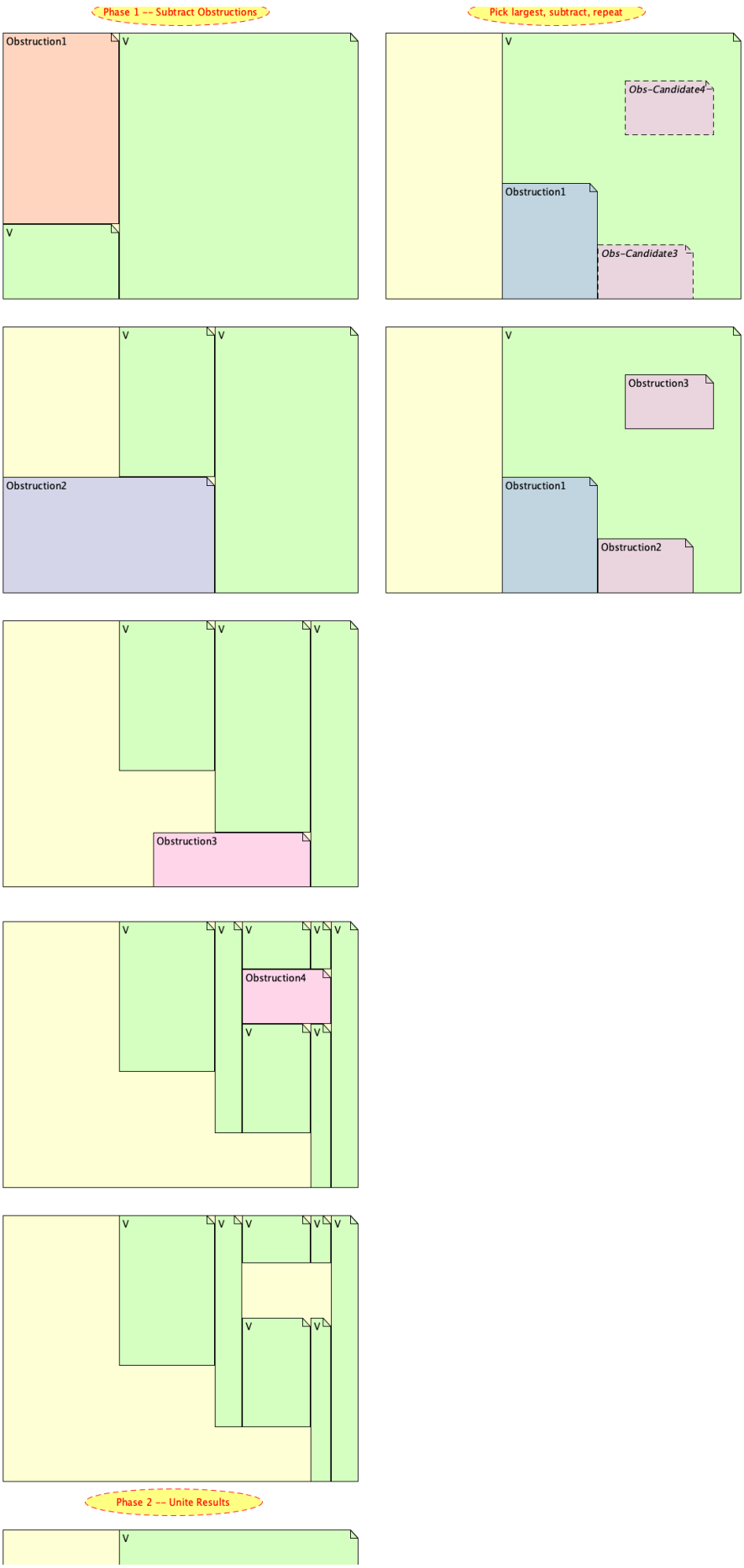


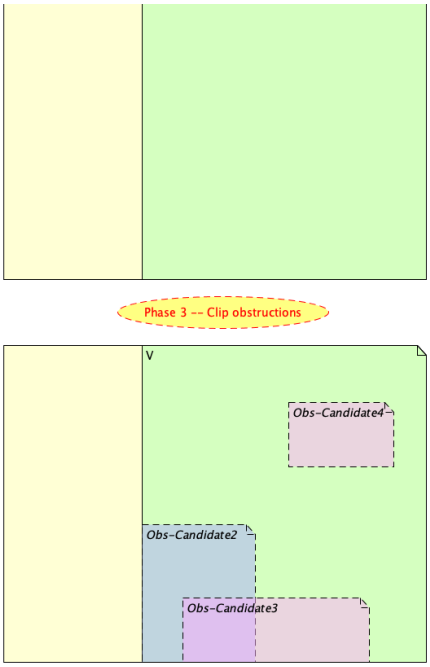
- (BOOL)collapseBoundingBox

Subtracts obstructions from clippedRect, then joins remains, and clips obstructions:



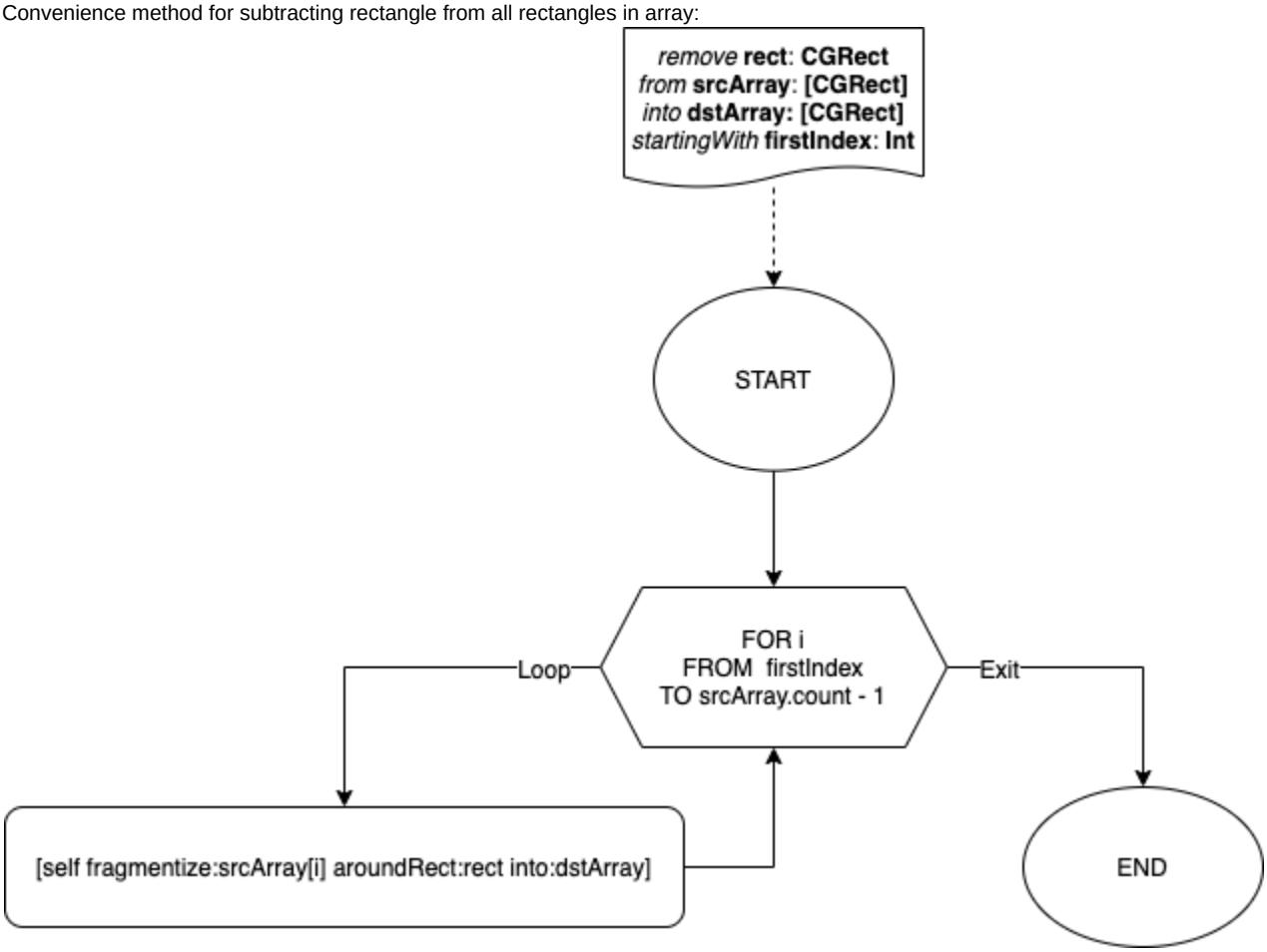



Phase 1 -- Subtract Obstructions
Pick largest, subtract, repeat
Obstruction1
V
V
V
Obs-Candidate4
Obstruction1
Obs-Candidate3
V
V
Obstruction2
V
Obstruction3
Obstruction1
Obstruction2
V
V
V
Obstruction3
V
V
V
V
V
V
Obstruction4
V
V
V
V
V
V
V
V
V
V
V
Phase 2 -- Unite Results
V



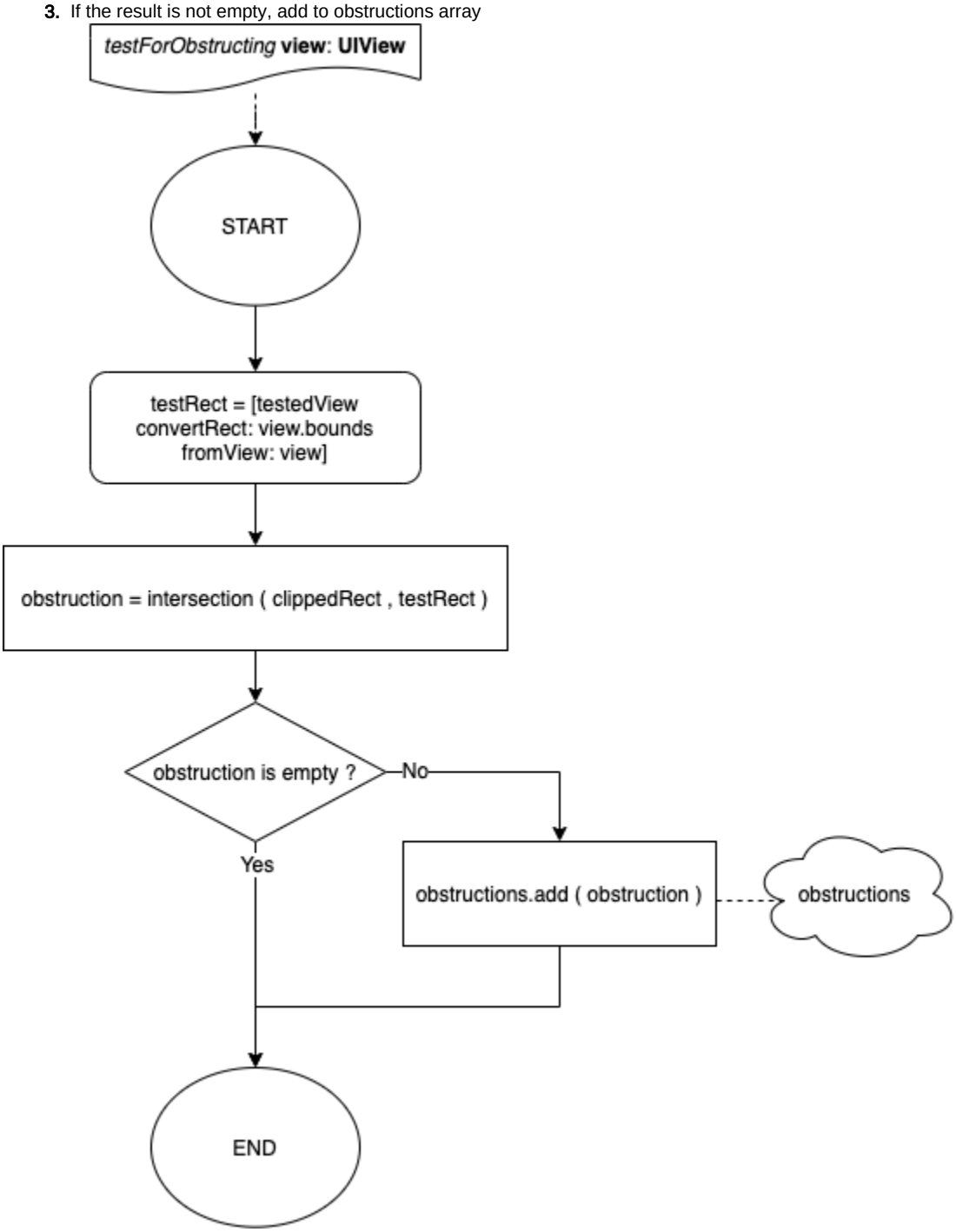
- (void)removeRect:(CGRect)rect from:(NSArray<NSValue *> *)srcArray into:(NSMutableArray<NSValue *> *)dstArray startingWith:(NSUInteger)firstIndex

Convenience method for subtracting rectangle from all rectangles in array:



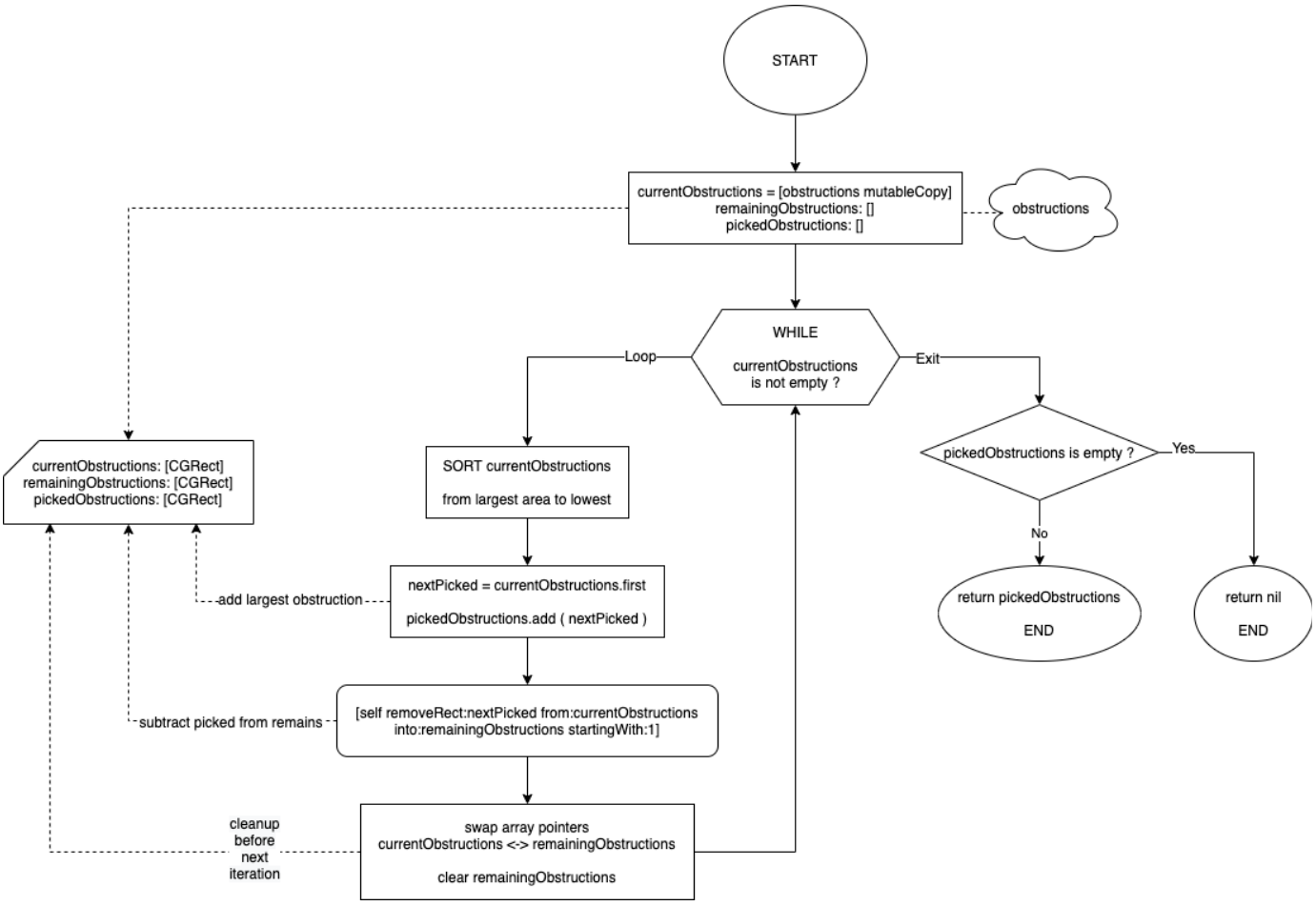
- (void)testForObstructing:(UIView *)view

1. Calculate projection of view onto testedView
2. Intersect with clippedRect
3. If the result is not empty, add to obstructions array



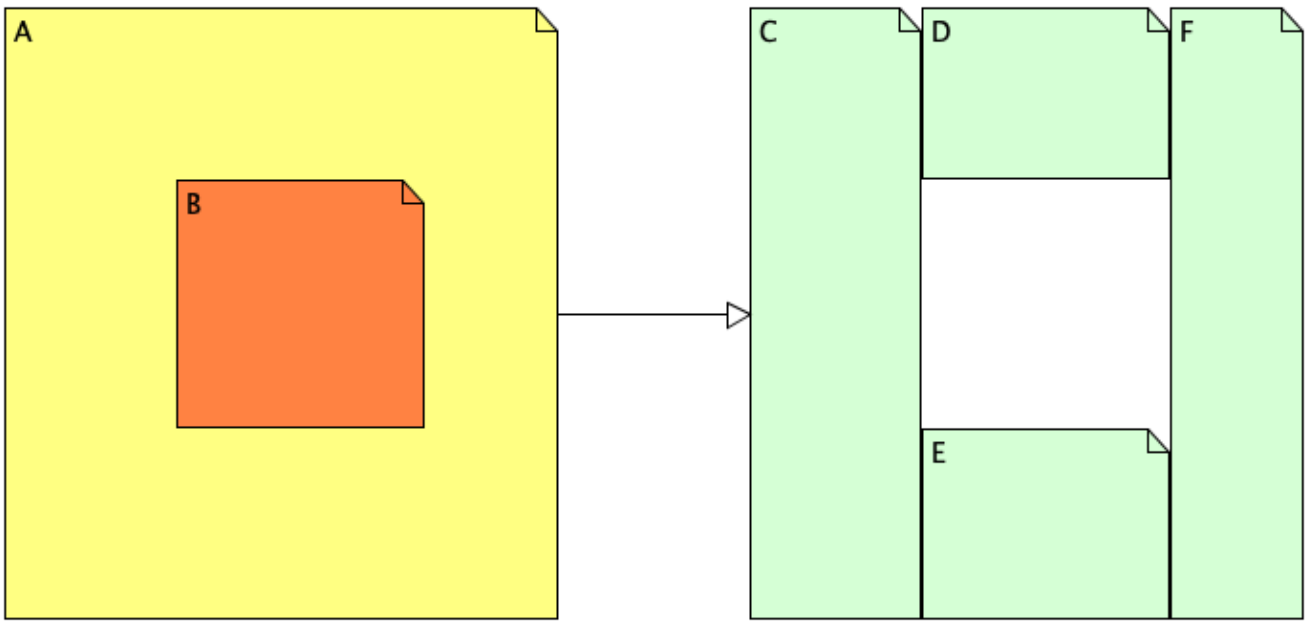
- (NSArray<NSValue *> *)buildObstructionRects

Calculates an array of non-intersecting obstruction rectangles, sorted from largest to lowest:



- (void)fragmentize:(NSValue *)value aroundRect:(CGRect)rect into:(NSMutableArray<NSValue *> *)array

Calculates the difference between two rectangles – results in 0-4 new rectangles – and add them into array:



1. If B does not intersect A
   - add B into output array, and exit

2. If B contains A:
   – exit
3. Project B onto A
4. If the projection is not empty:
   - Create an array of 4 rectangles (C, D, E, F)
   - Add all non-empty rectangles into output array