48th International Symposium on Microarchitecture

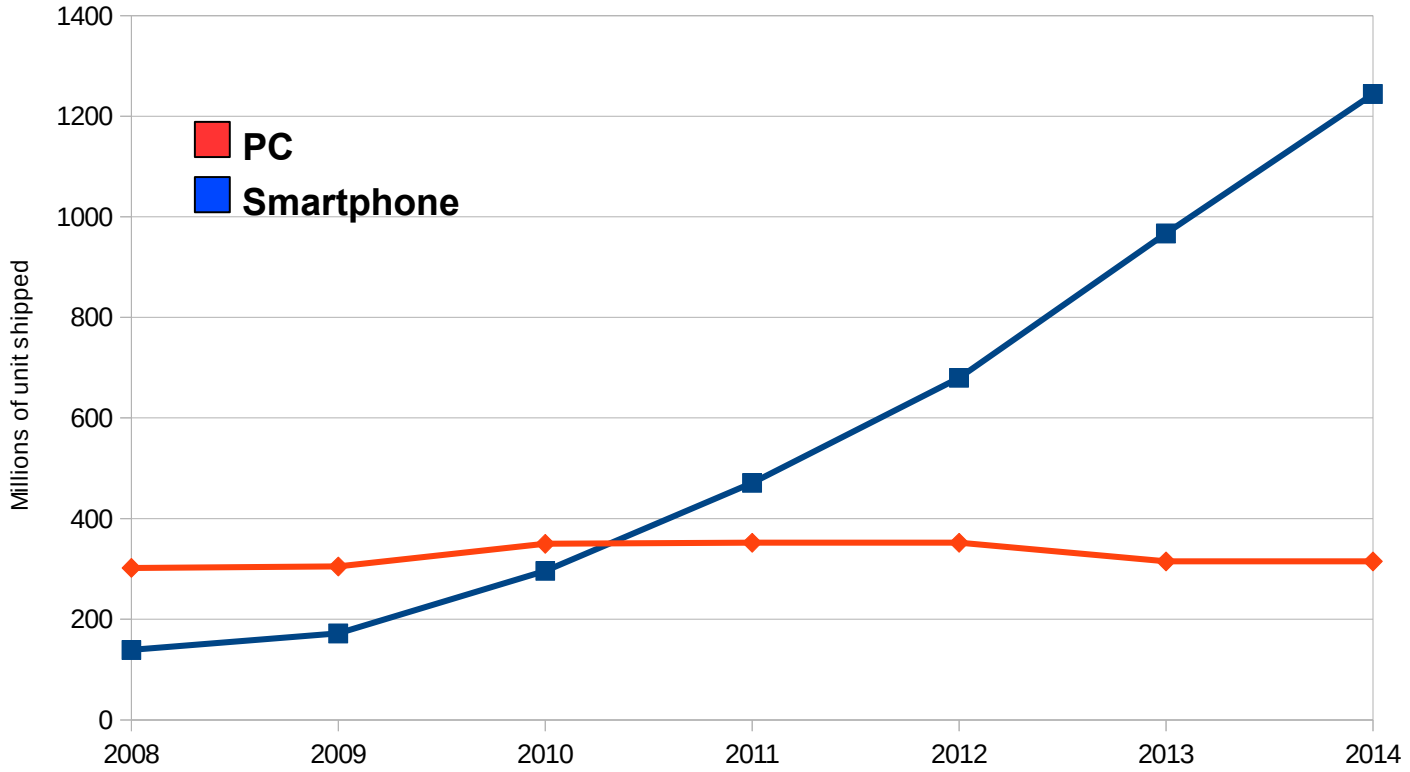
Ultra-Low Power Render-Based Collision Detection for CPU/GPU Systems

Enrique de Lucas
Joan-Manuel Parcerisa
Pedro Marcuello
Antonio González

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Market

Mobile devices market is growing fast



Mobile Systems

- Users demand **realistic** and **complex** graphics like in laptops and desktops
  - **Battery life** is about 4 hours for GFXBench 3.0![1]
  - **Heat dissipation**

- "**CPU**, **GPU** and screen are the dominant energy consumers on a smartphone"[2]
- Graphics animation applications are **quite popular**
  - Collision Detection is an important task

1. With an ARM Mali 400MP GPU, www.gfxbench.com
2. Mittal et al., Empowering Developers to Estimate App Energy Consumption, Aug. 2012.

Outline

1. Motivation
2. **Collision Detection (CD)**
3. Render-Based CD in the GPU
4. Results
5. Conclusions

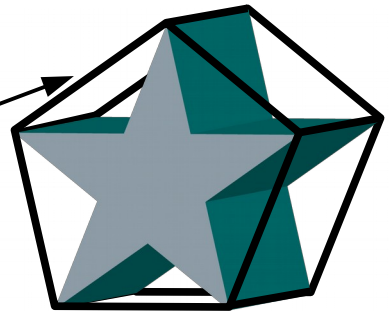
Collision Detection (CD)

Frame *i*

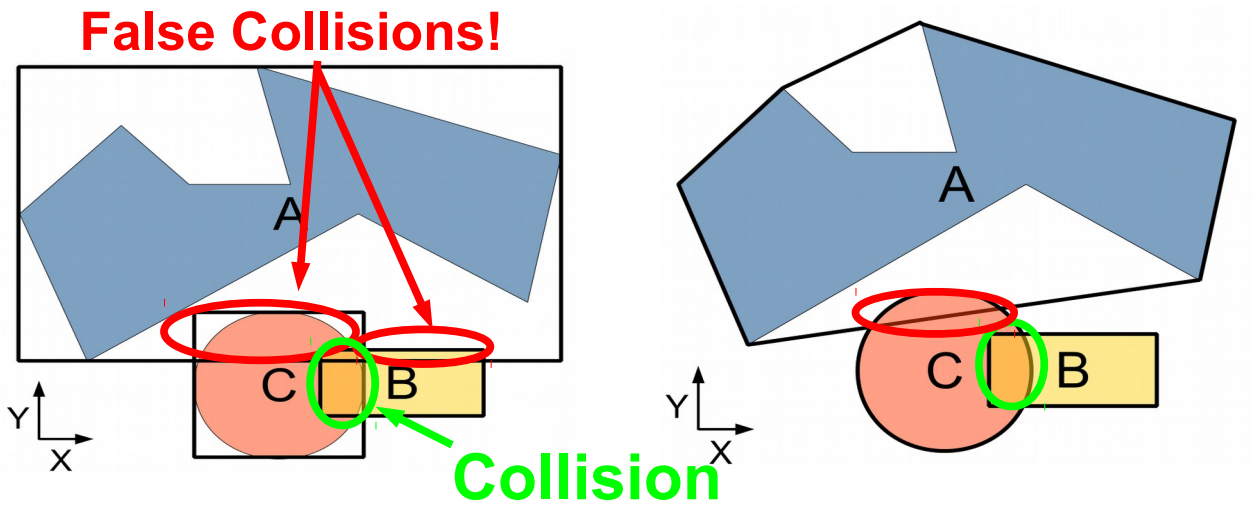
Frame *i+2*

CD identifies the contact points between objects

Bounding Volume



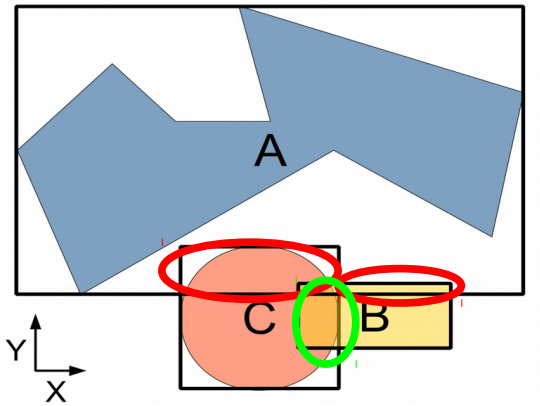
Bounding Volumes

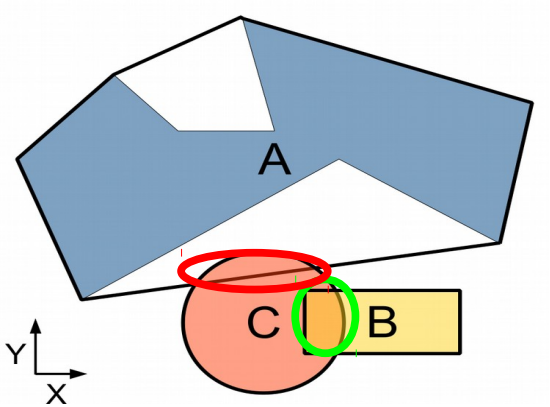


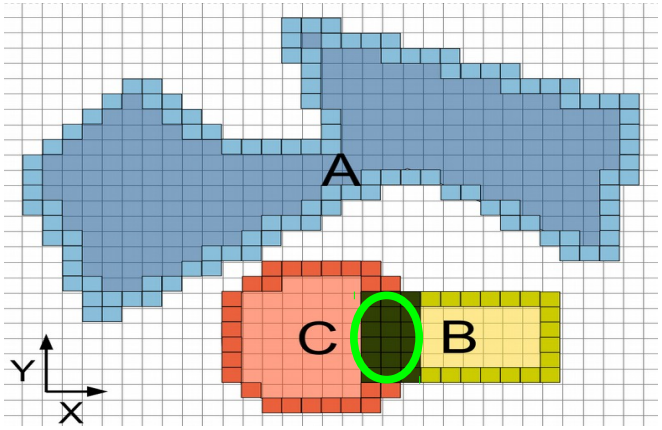
Bounding Volume	Cuboid	Convex Hull
Accuracy	Low	Medium
Computing Cost	Low	High

false area = false collisions

Image-Based CD (IBCD)

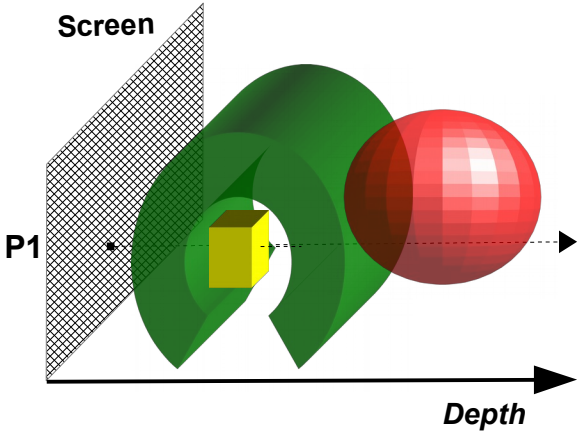






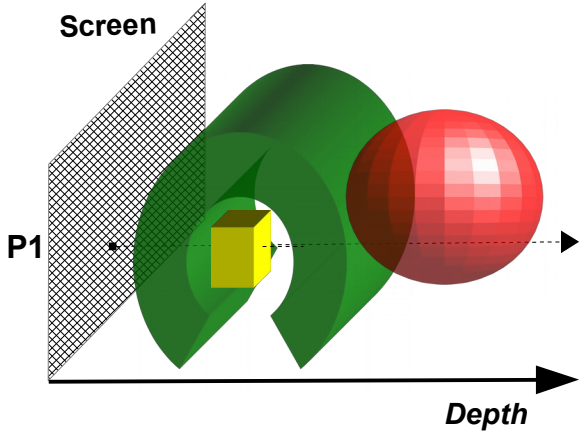
- CD performed at **pixel granularity**
- No Bounding Volumes
- Higher Accuracy
- Computing/Energy Cost
  - CPU: **huge**
  - Our technique (GPU): **tiny**

How does IBCD work?



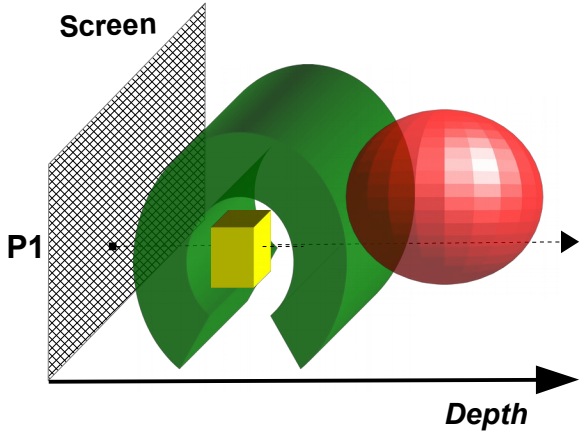
Steps of Image-Based CD

How does IBCD work?



Steps of Image-Based CD

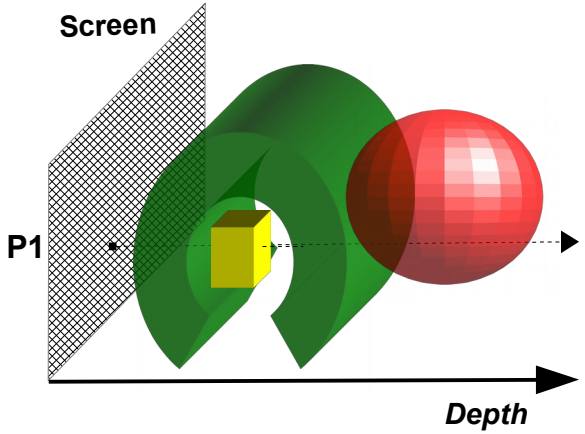
How does IBCD work?

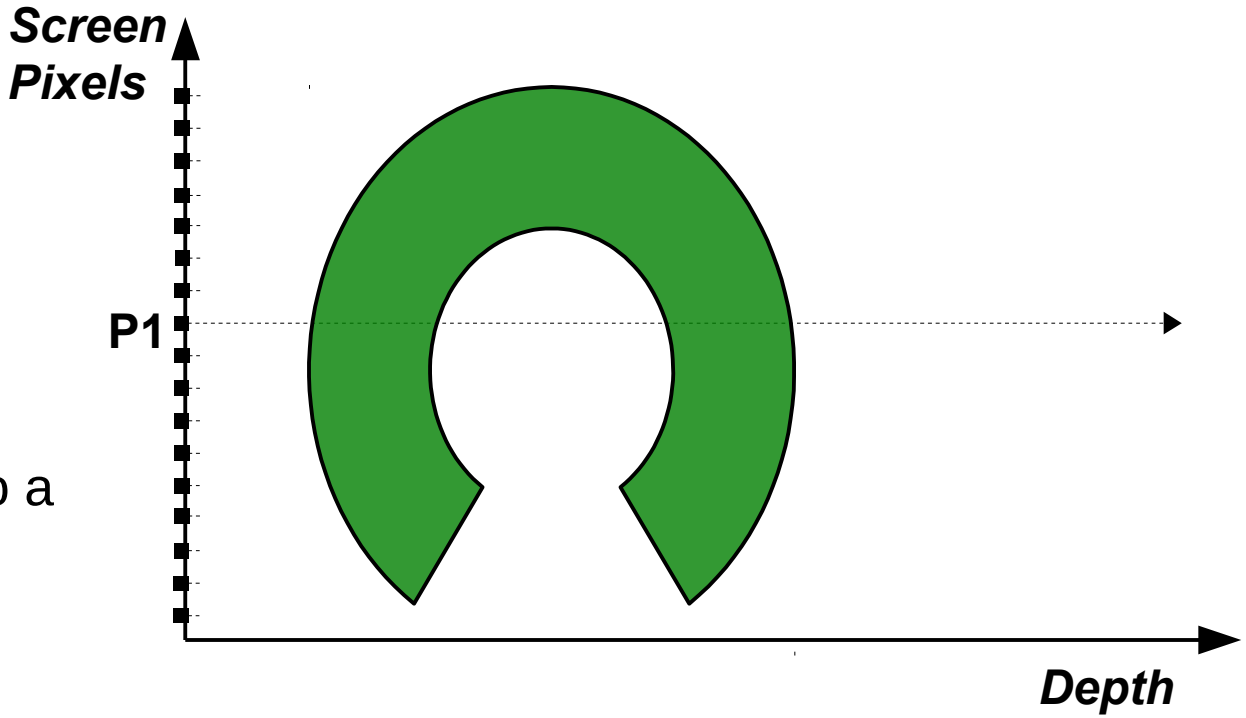


Steps of Image-Based CD

1) **Project** objects onto a plane (screen)

How does IBCD work?

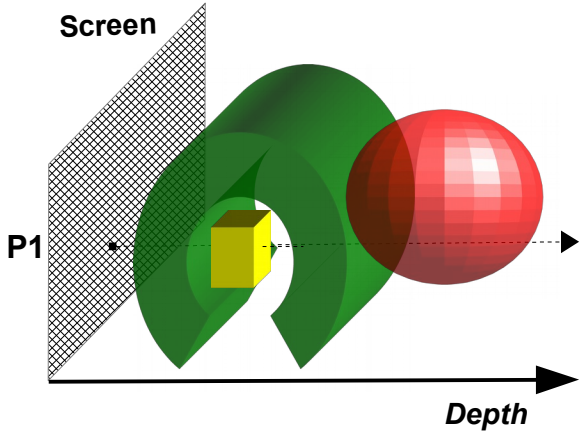


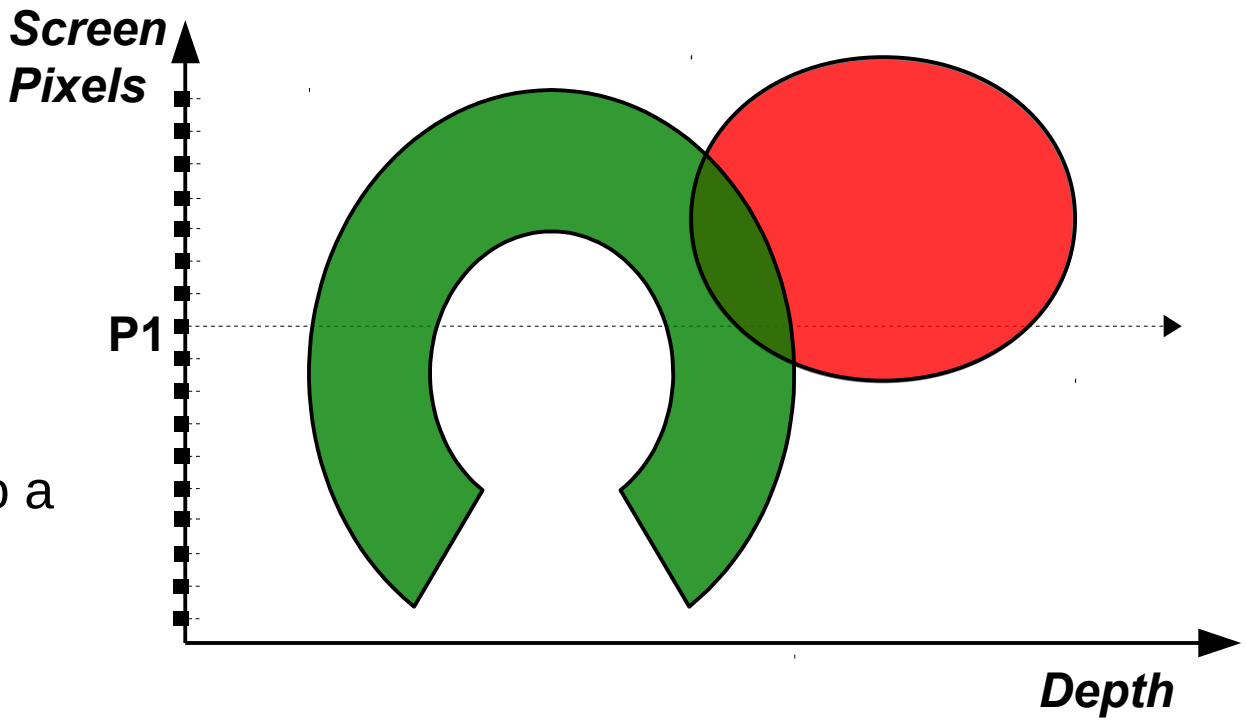


Steps of Image-Based CD

1) **Project** objects onto a plane (screen)

How does IBCD work?

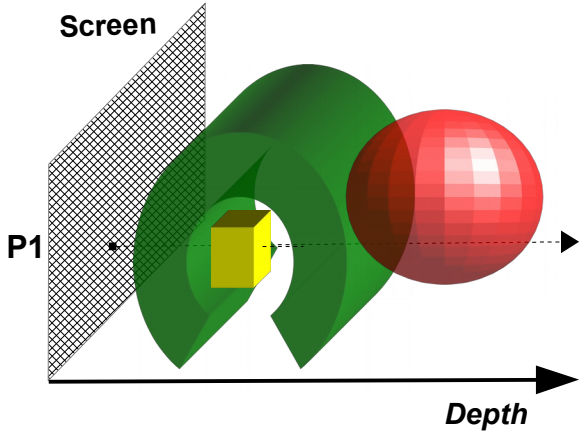


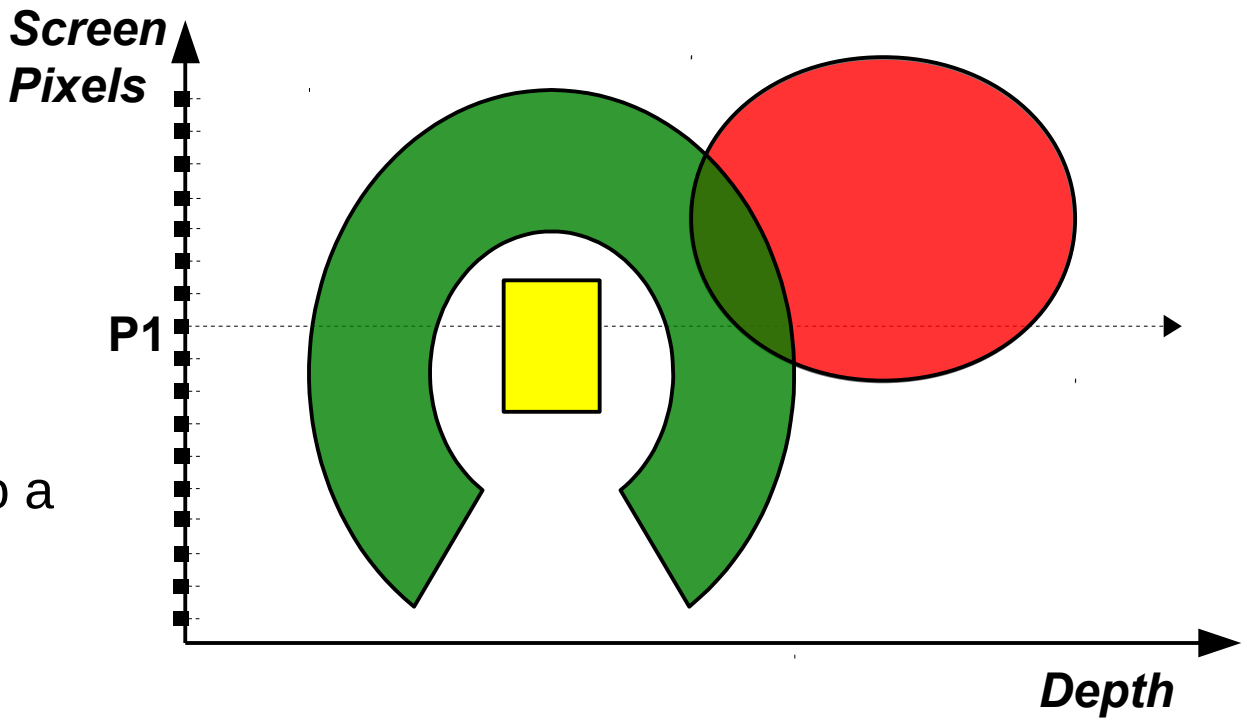


Steps of Image-Based CD

1) **Project** objects onto a plane (screen)

How does IBCD work?

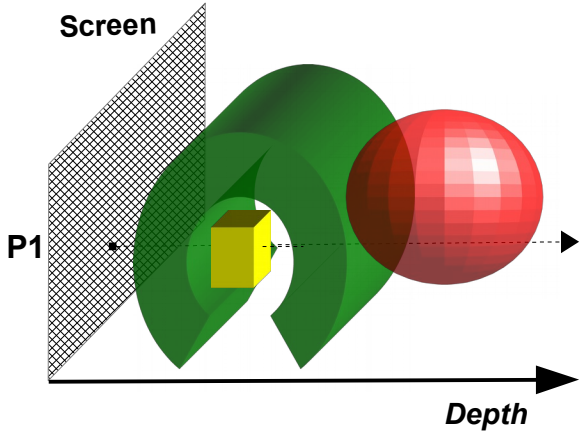




Steps of Image-Based CD

1) **Project** objects onto a plane (screen)

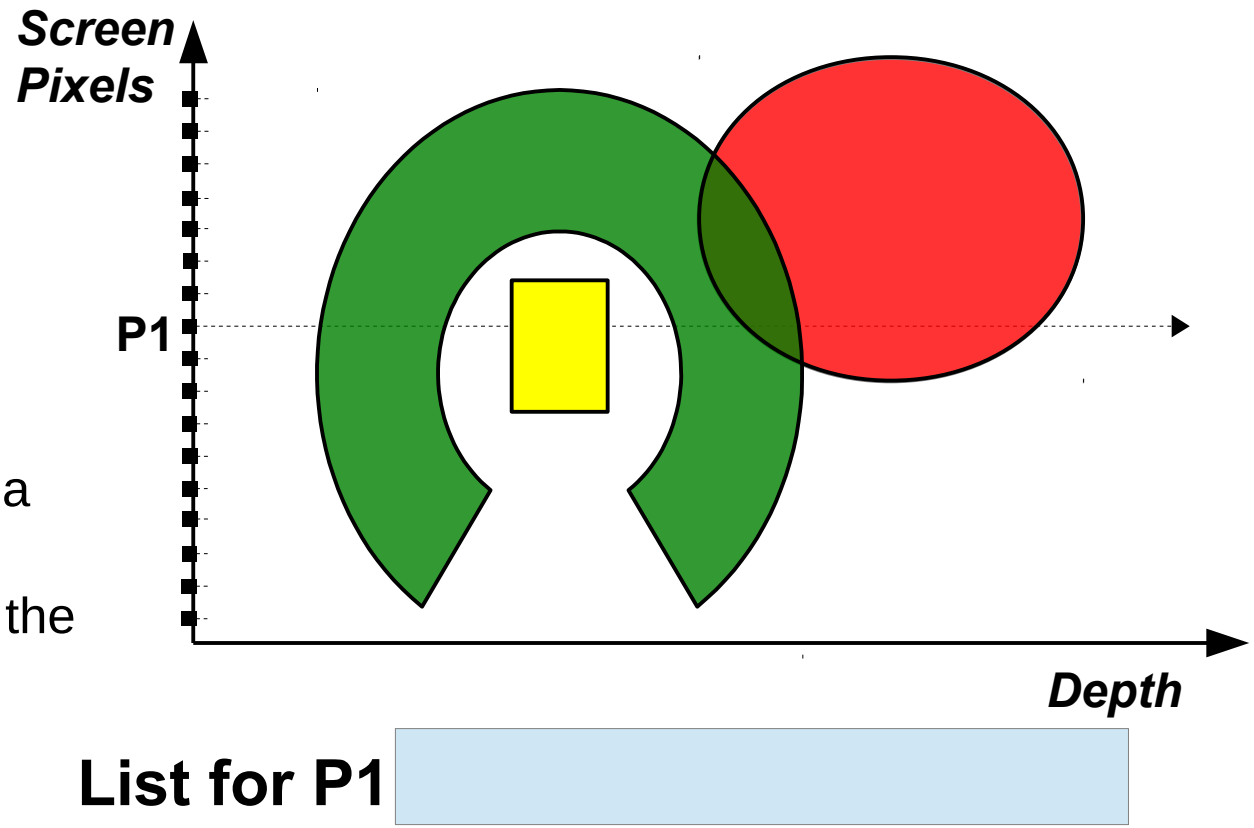
How does IBCD work?



Steps of Image-Based CD

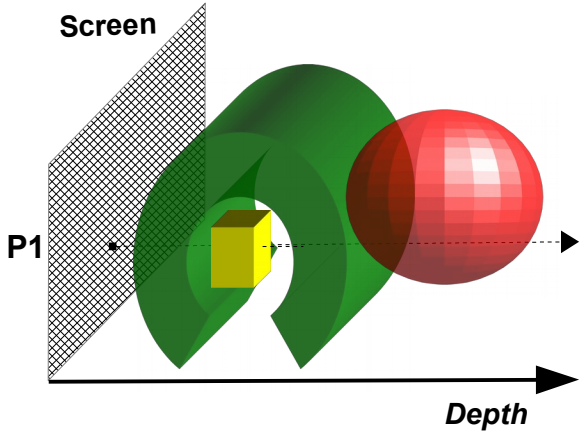
1) **Project** objects onto a plane (screen)

2) **Rasterize** surface of the objects and store depths in a **list**



List for P1

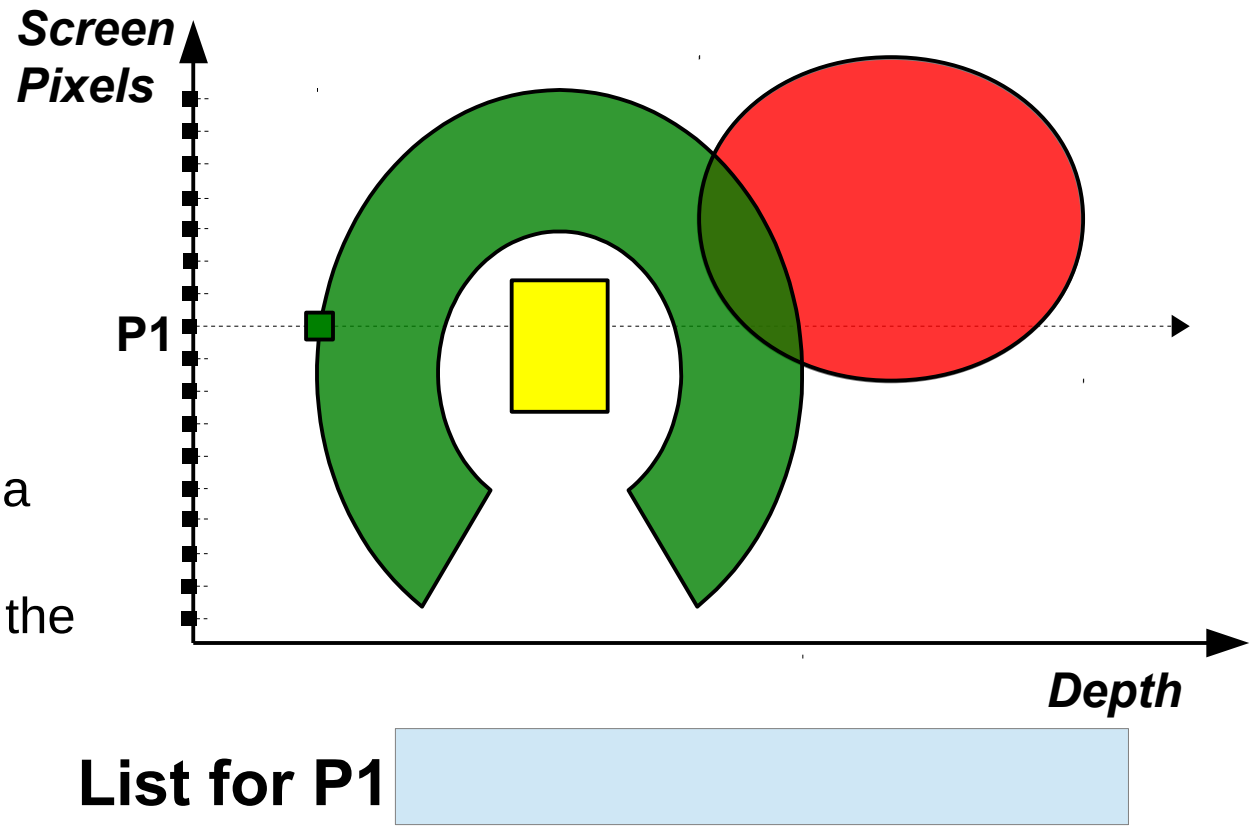
How does IBCD work?



Steps of Image-Based CD

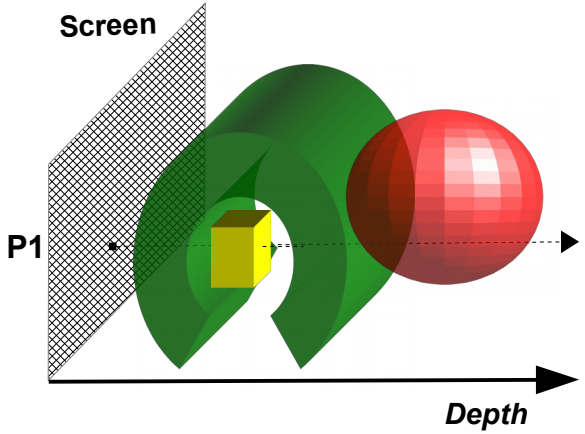
1) **Project** objects onto a plane (screen)

2) **Rasterize** surface of the objects and store depths in a **list**



List for P1

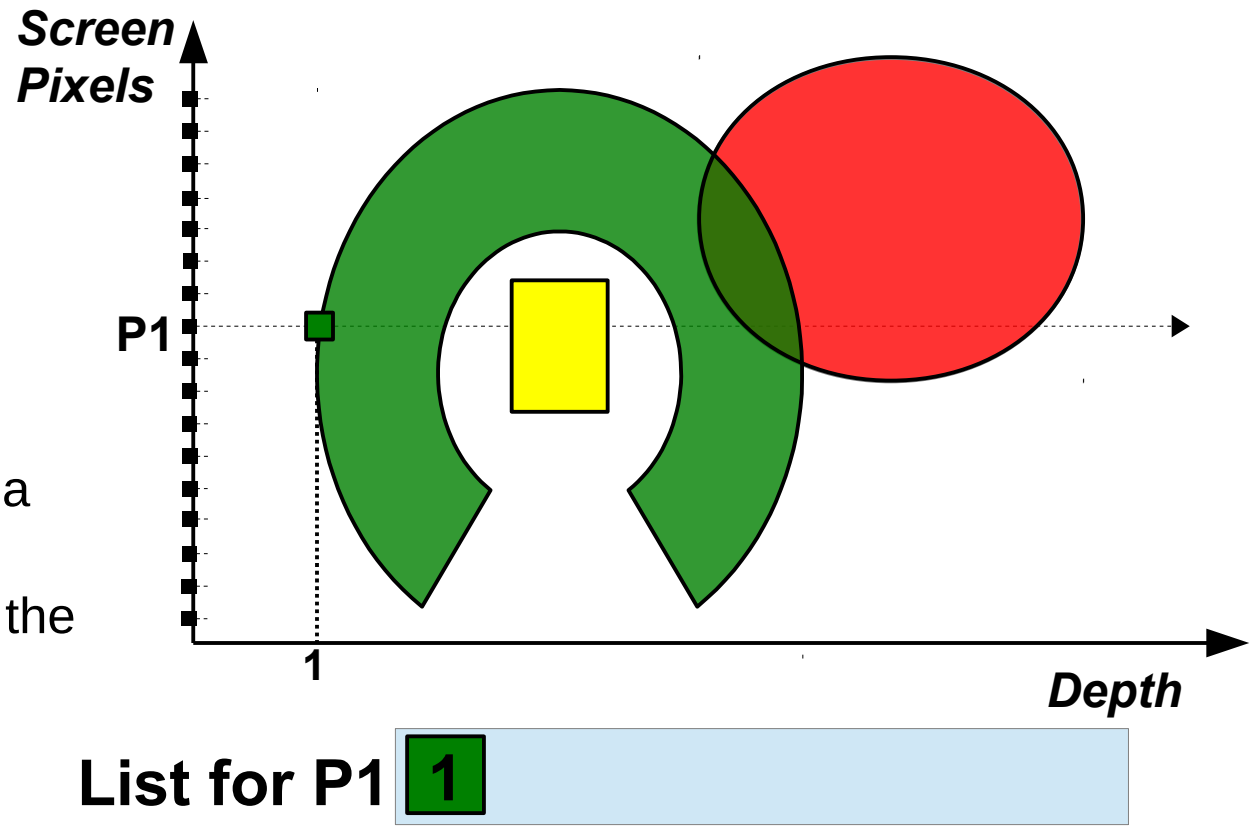
How does IBCD work?



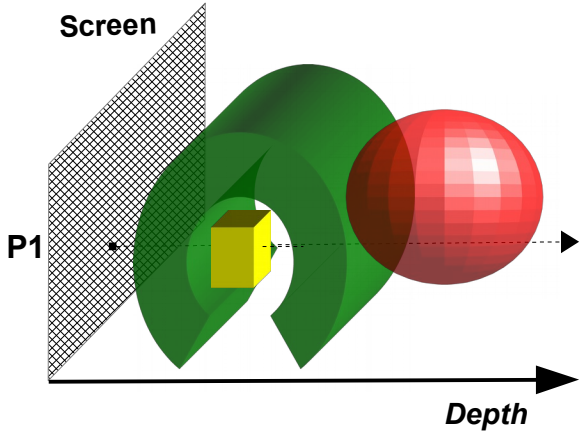
Steps of Image-Based CD

1) **Project** objects onto a plane (screen)

2) **Rasterize** surface of the objects and store depths in a **list**



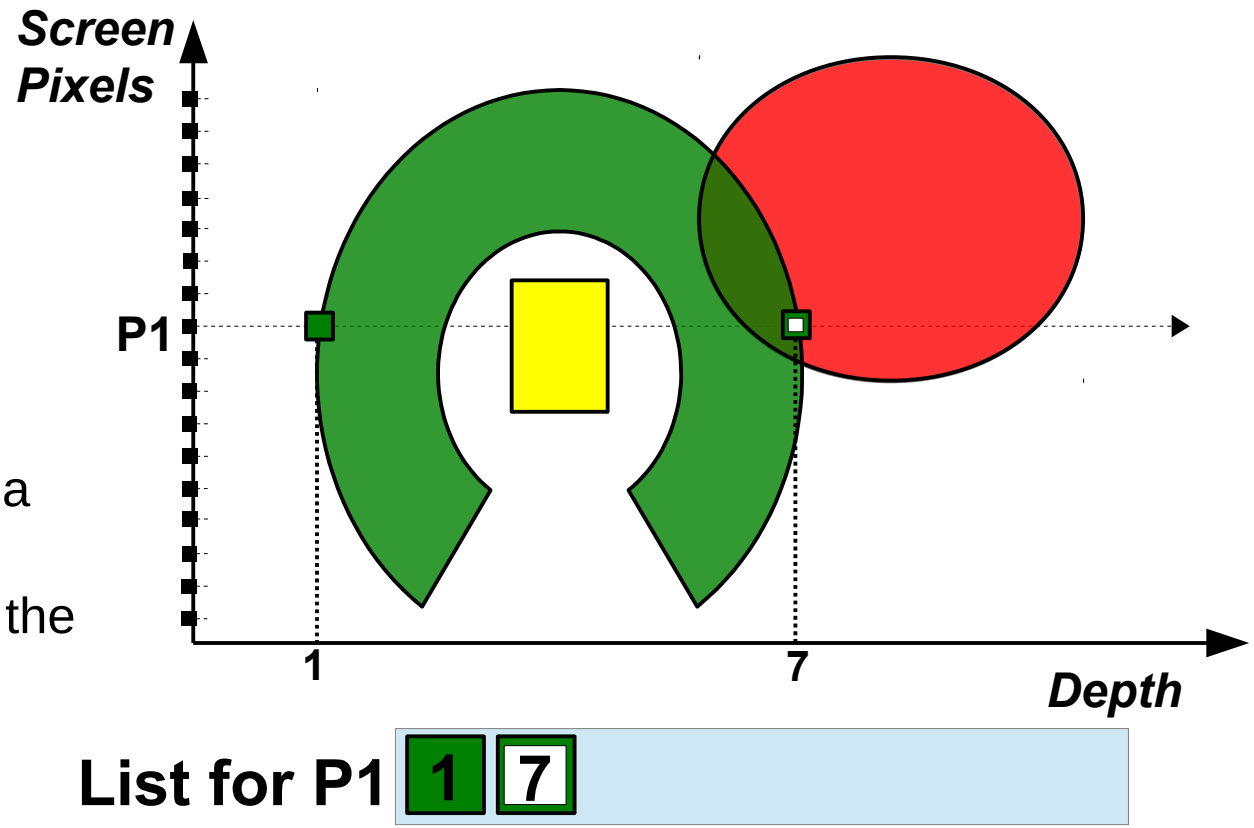
How does IBCD work?



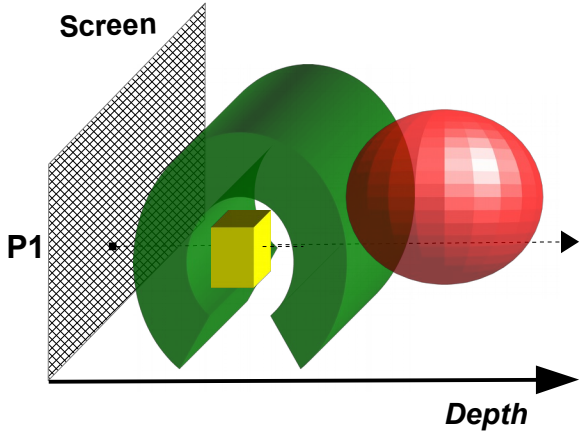
Steps of Image-Based CD

1) **Project** objects onto a plane (screen)

2) **Rasterize** surface of the objects and store depths in a **list**



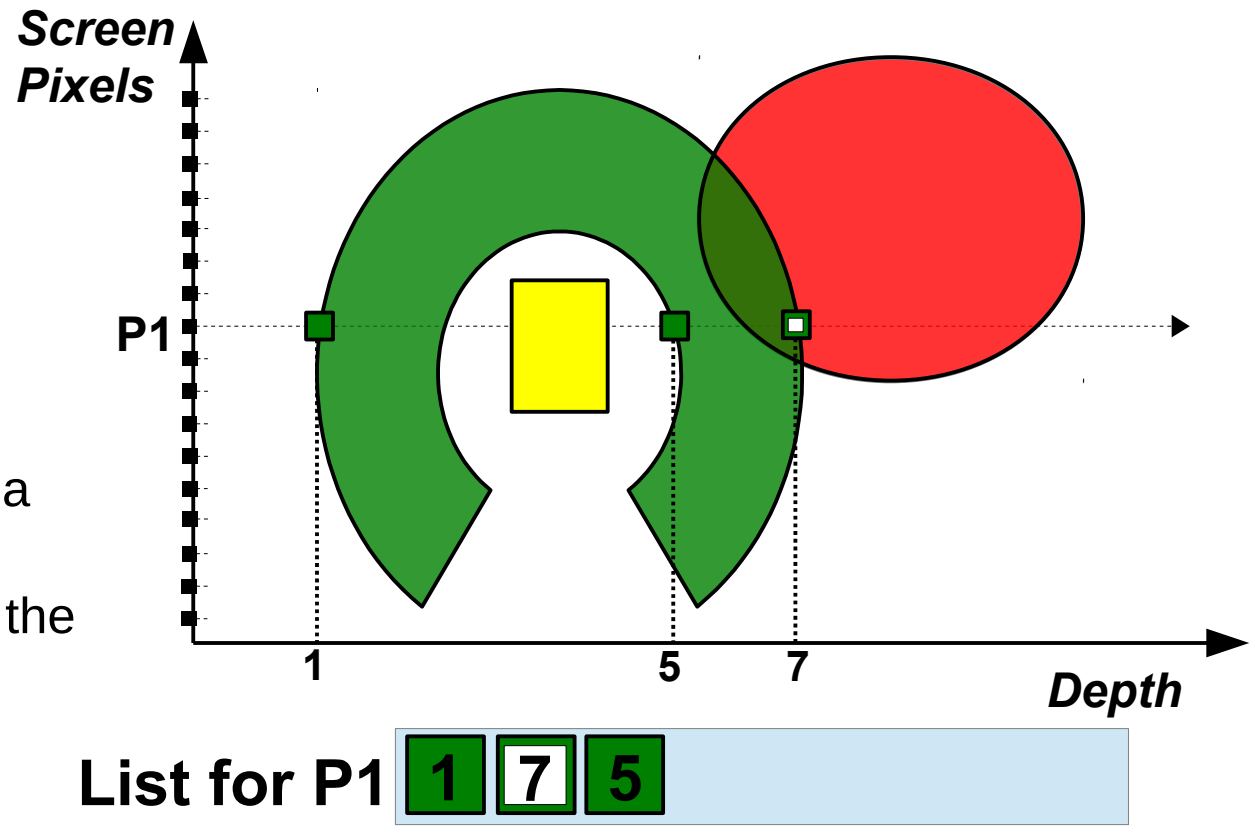
How does IBCD work?



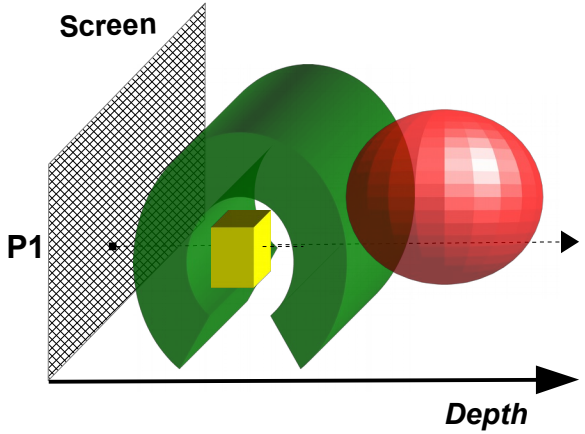
Steps of Image-Based CD

1) **Project** objects onto a plane (screen)

2) **Rasterize** surface of the objects and store depths in a **list**



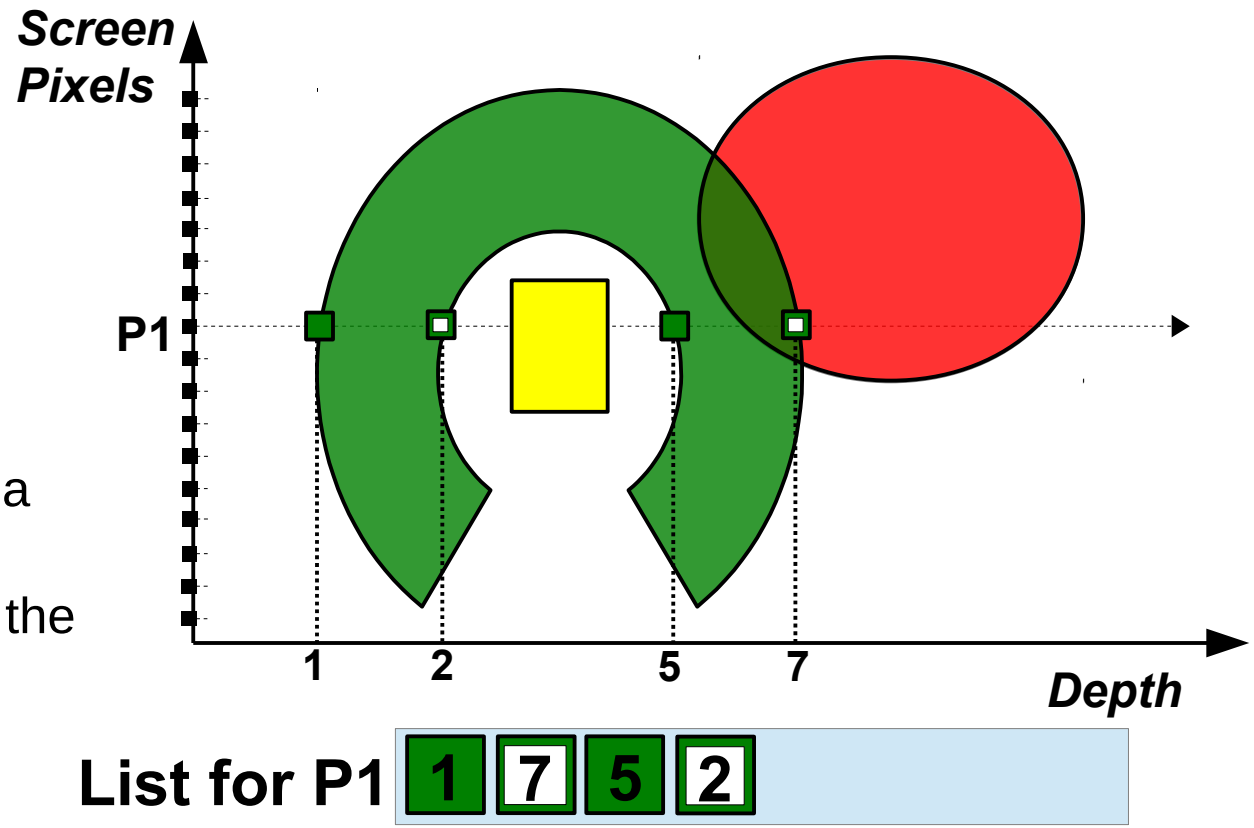
How does IBCD work?



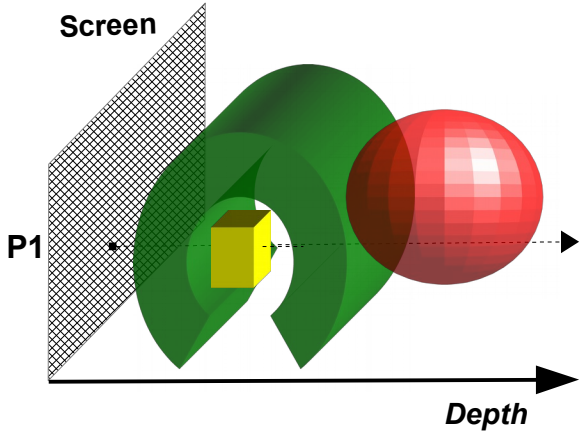
Steps of Image-Based CD

1) **Project** objects onto a plane (screen)

2) **Rasterize** surface of the objects and store depths in a **list**



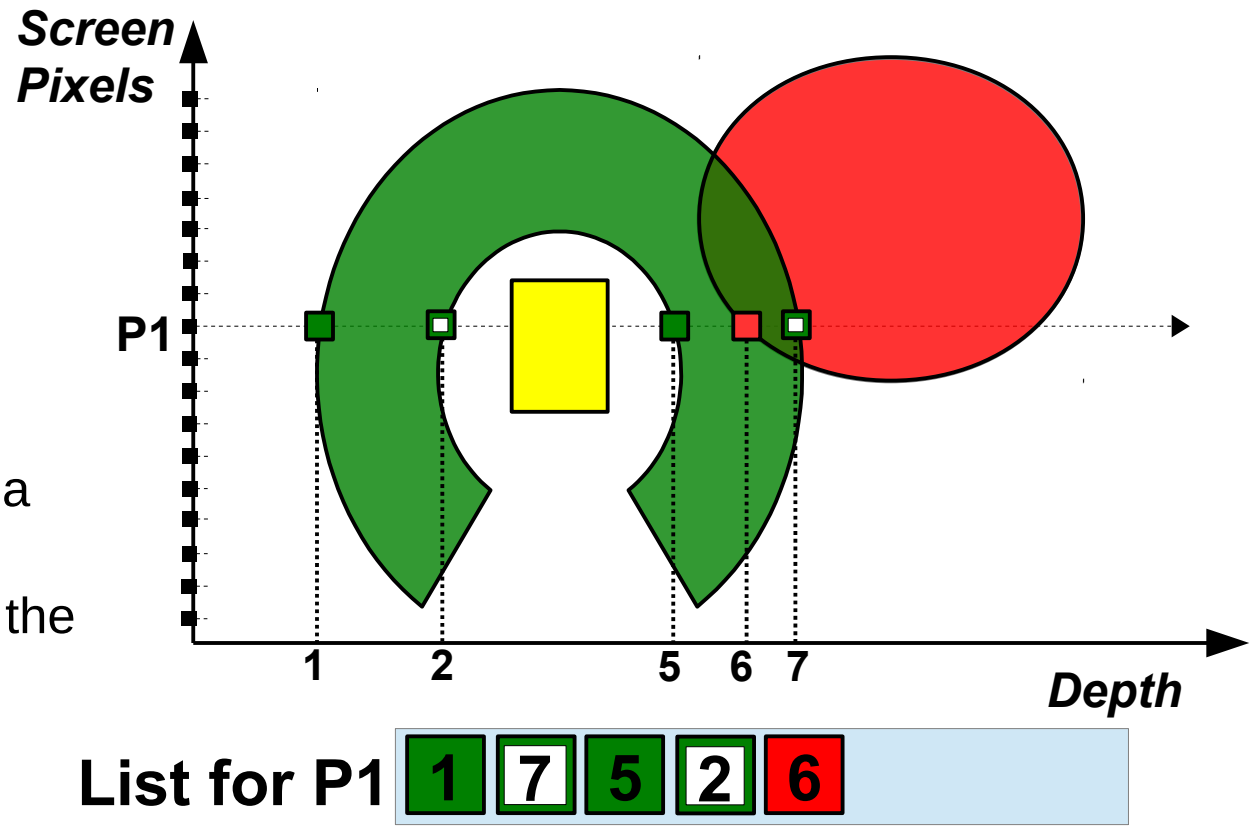
How does IBCD work?



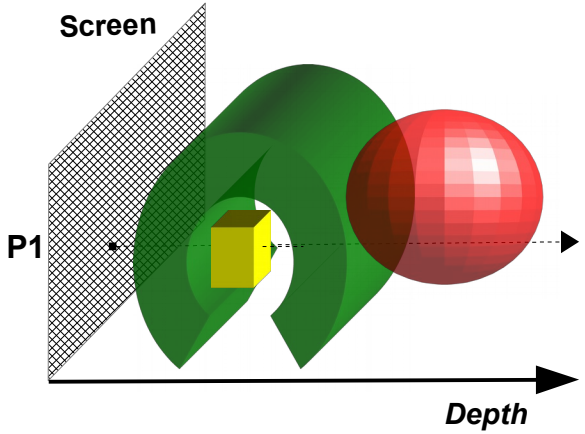
Steps of Image-Based CD

1) **Project** objects onto a plane (screen)

2) **Rasterize** surface of the objects and store depths in a **list**



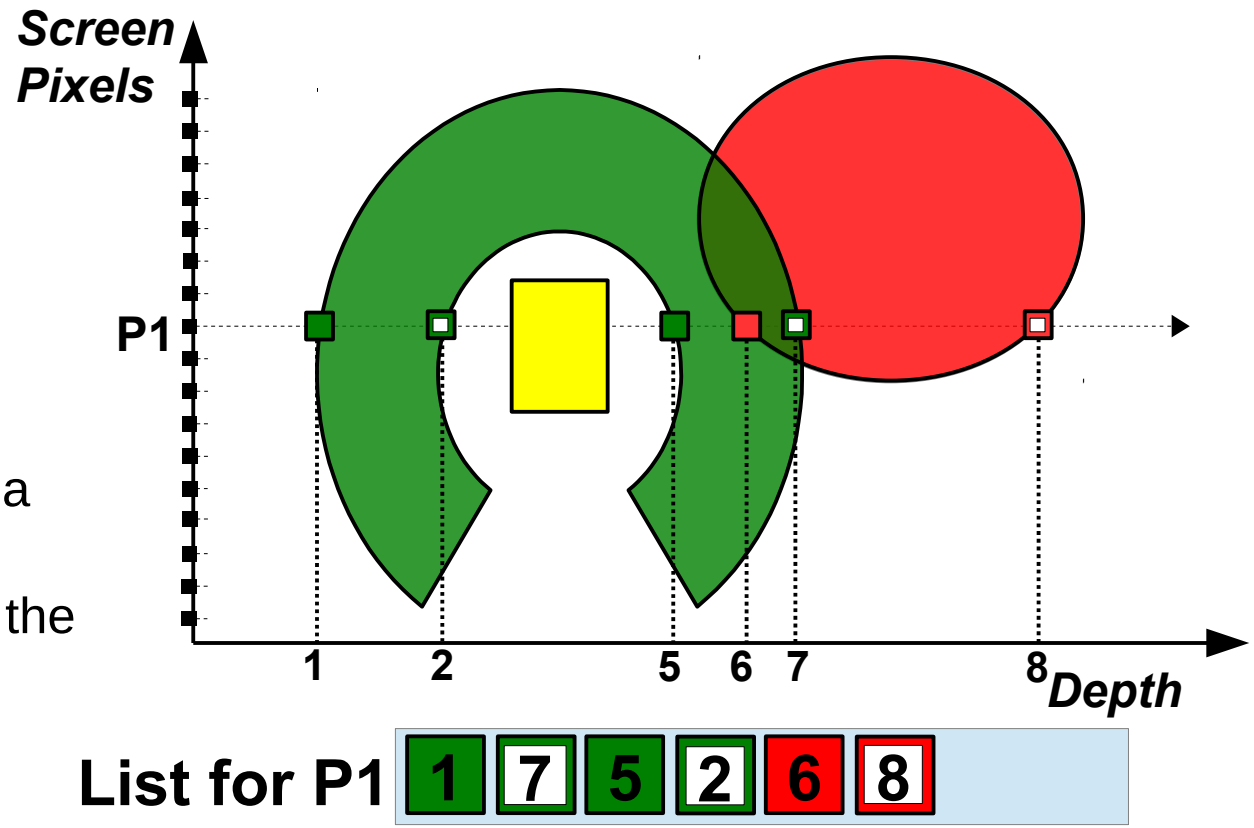
How does IBCD work?



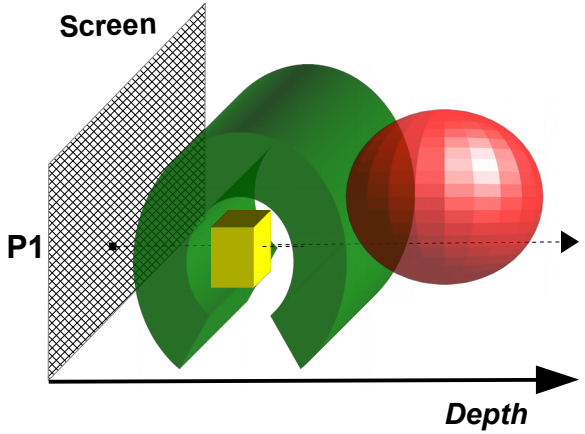
Steps of Image-Based CD

1) **Project** objects onto a plane (screen)

2) **Rasterize** surface of the objects and store depths in a **list**



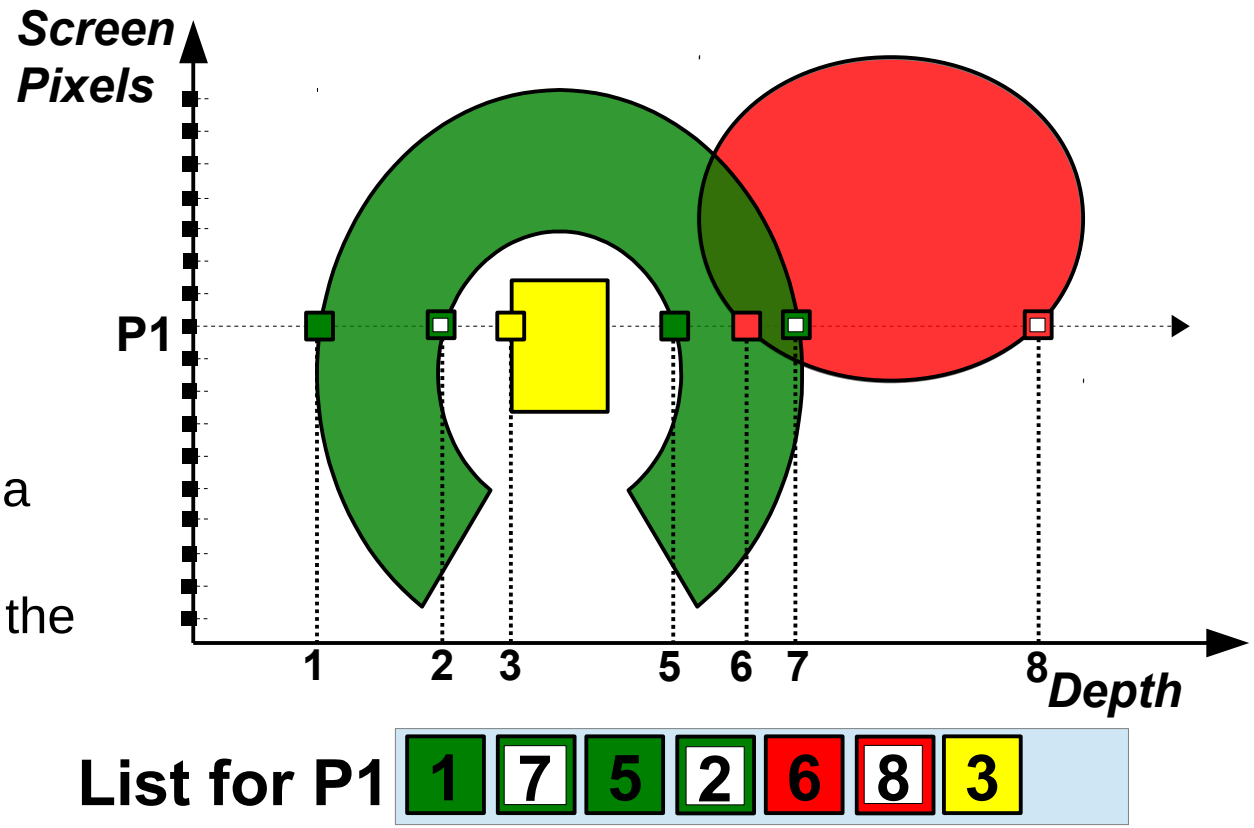
How does IBCD work?



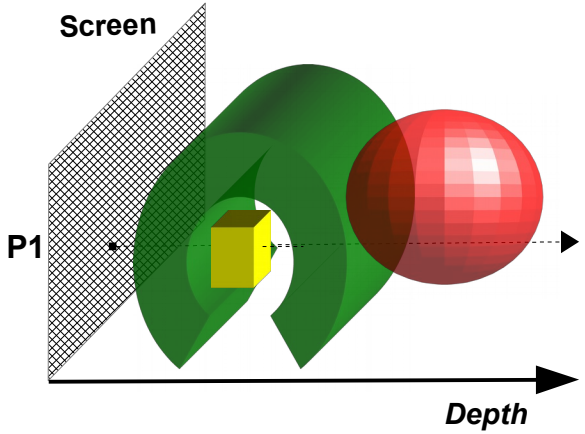
Steps of Image-Based CD

1) **Project** objects onto a plane (screen)

2) **Rasterize** surface of the objects and store depths in a **list**



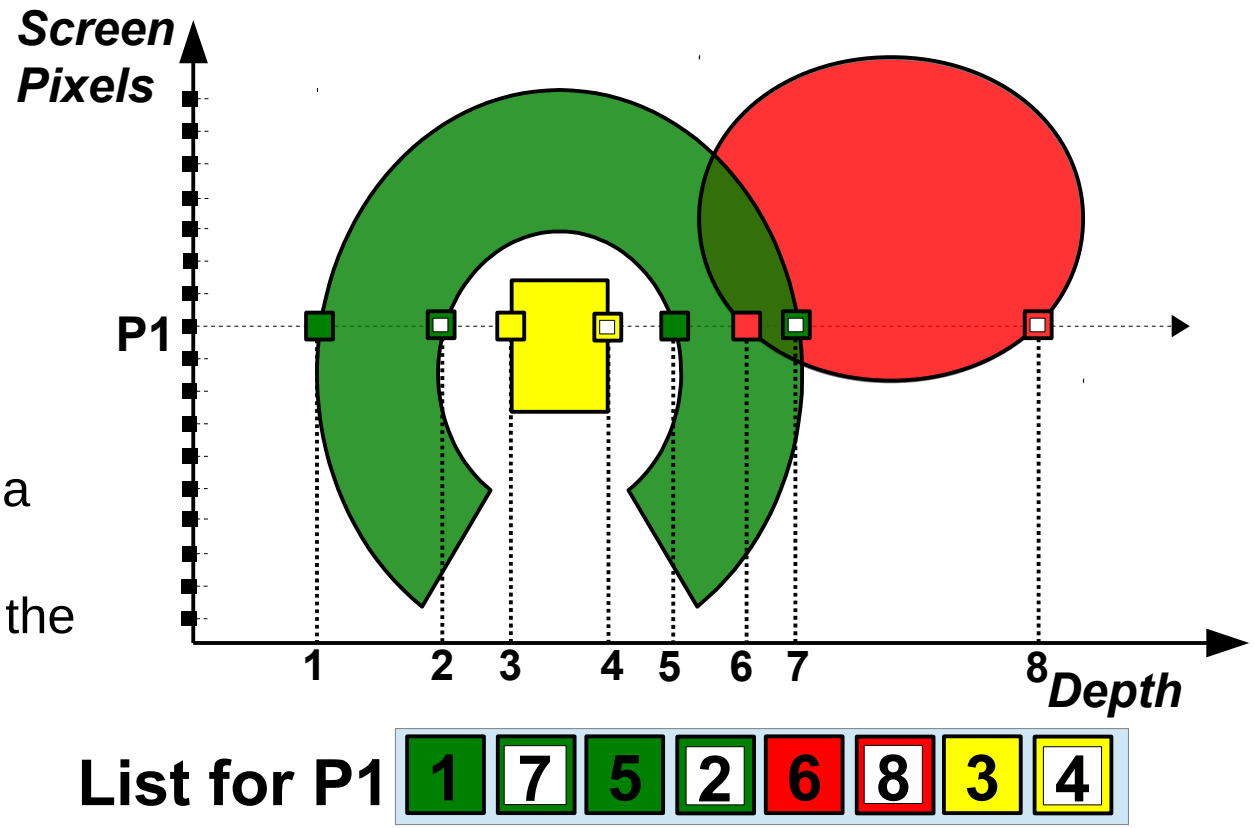
How does IBCD work?



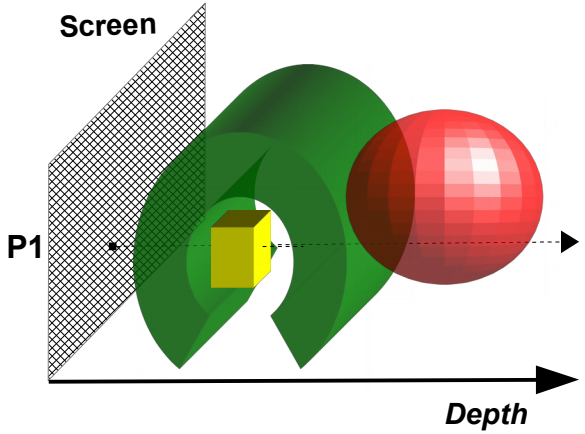
Steps of Image-Based CD

1) **Project** objects onto a plane (screen)

2) **Rasterize** surface of the objects and store depths in a **list**

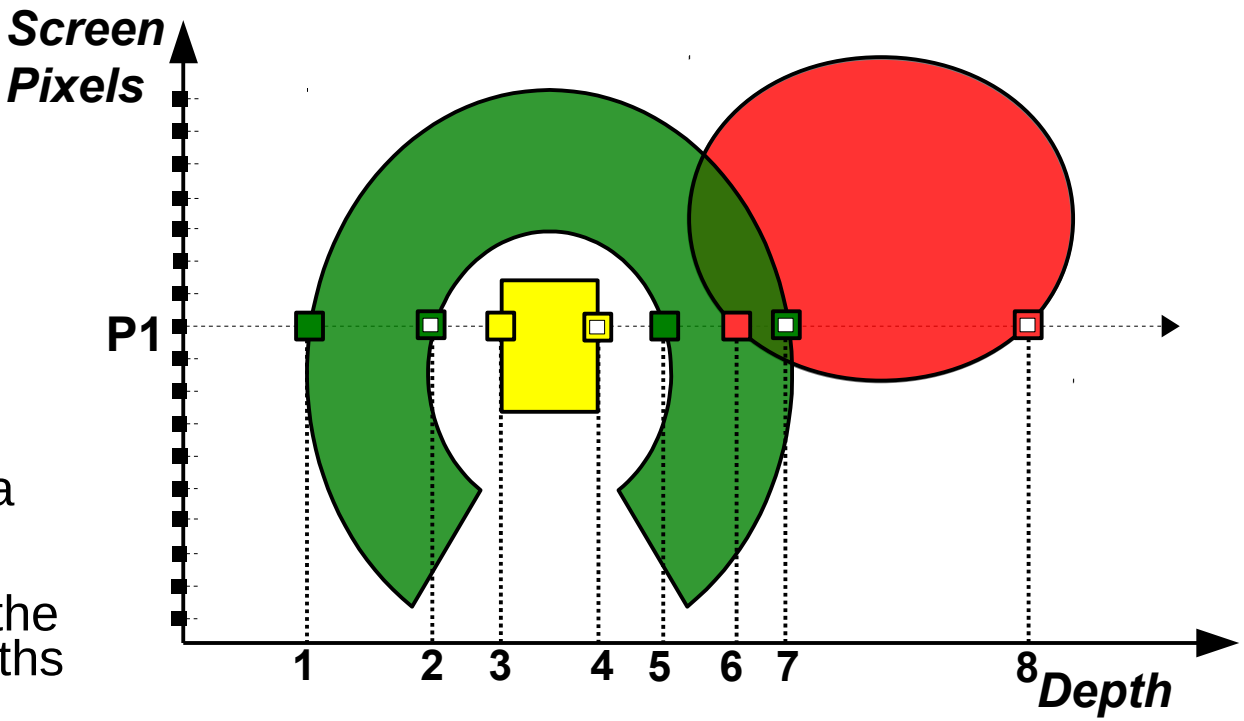


How does IBCD work?



Steps of Image-Based CD

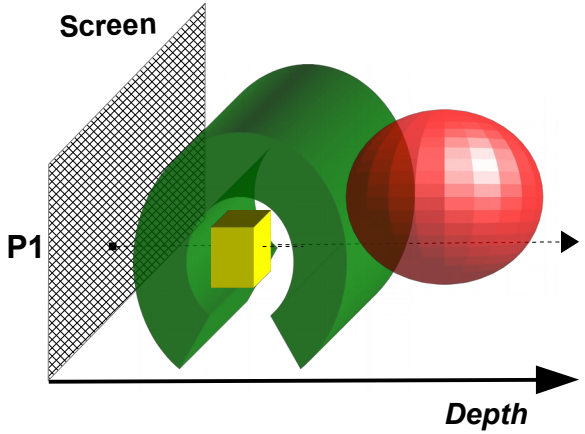
1) **Project** objects onto a plane (screen)
2) **Rasterize** surface of the objects and store depths in a **list**
3) **Sort** values by depth



List for P1 1 7 5 2 6 8 3 4

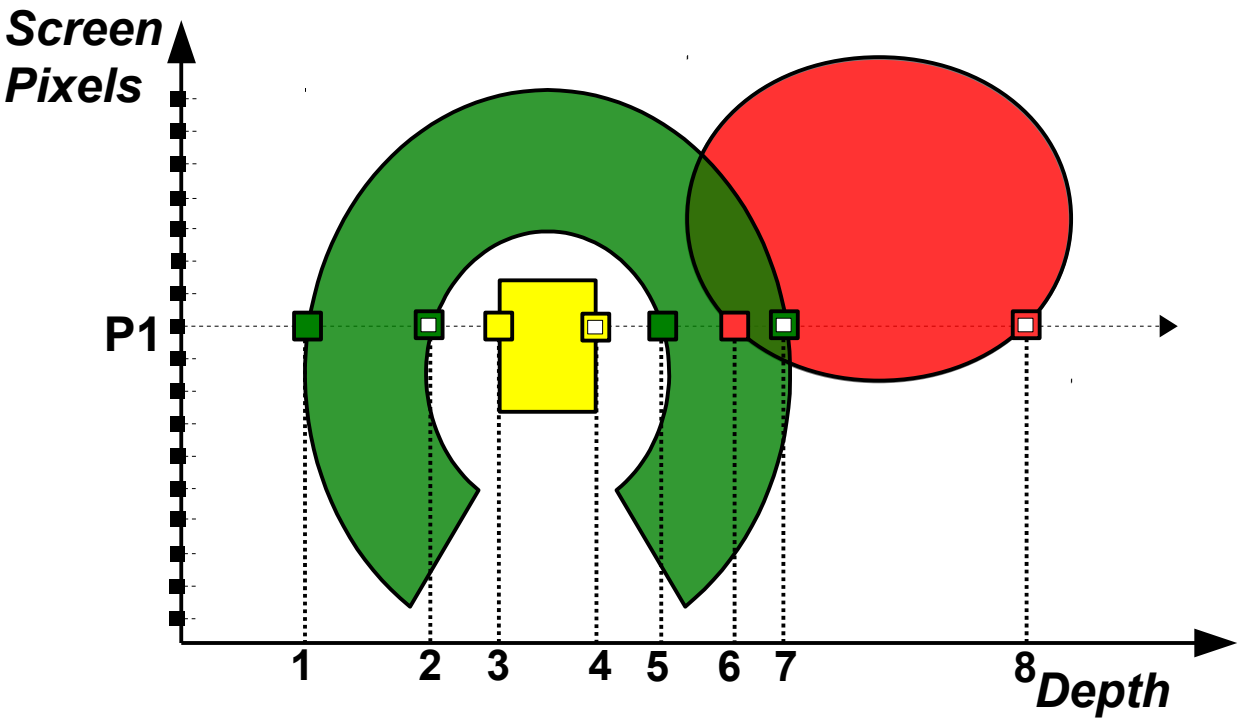
Sort 1 2 3 4 5 6 7 8

How does IBCD work?



Steps of Image-Based CD

1) **Project** objects onto a plane (screen)
2) **Rasterize** surface of the objects and store depths in a **list**
3) **Sort** values by depth
4) **Detect** overlapping depth-ranges

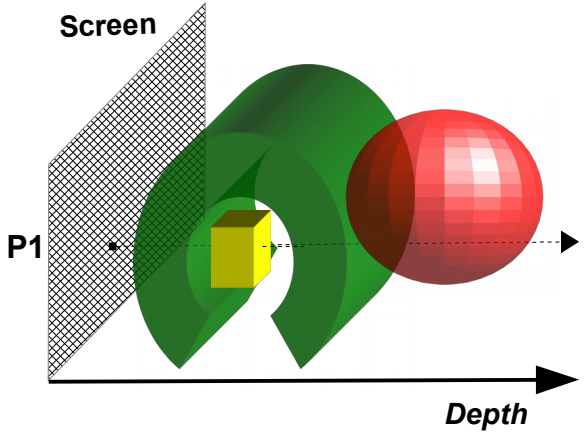


List for P1 1 7 5 2 6 8 3 4

Sort 1 2 3 4 5 6 7 8

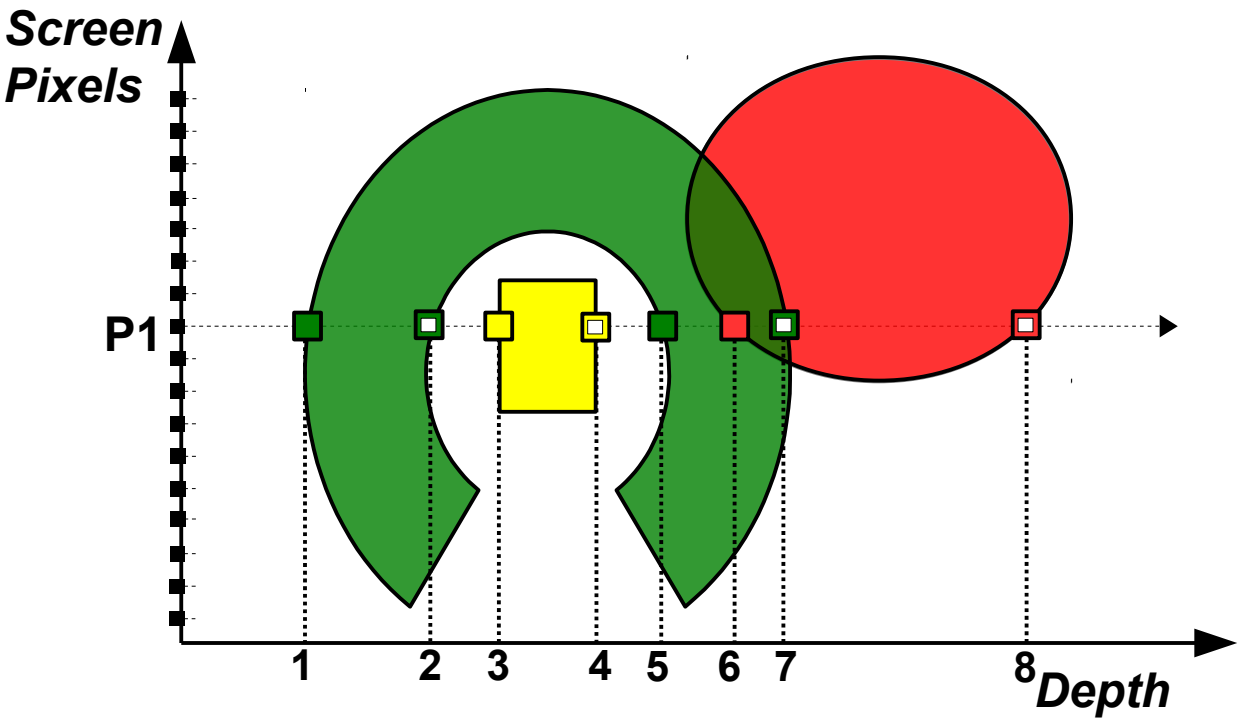
Depth-ranges

How does IBCD work?



Steps of Image-Based CD

1) **Project** objects onto a plane (screen)
2) **Rasterize** surface of the objects and store depths in a **list**
3) **Sort** values by depth
4) **Detect** overlapping depth-ranges

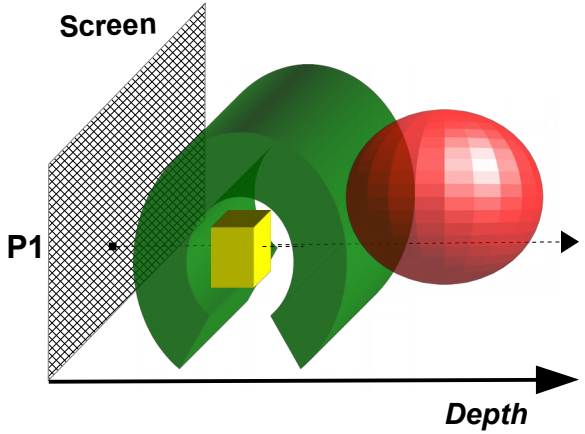


List for P1 1 7 5 2 6 8 3 4

Sort 1 2 3 4 5 6 7 8

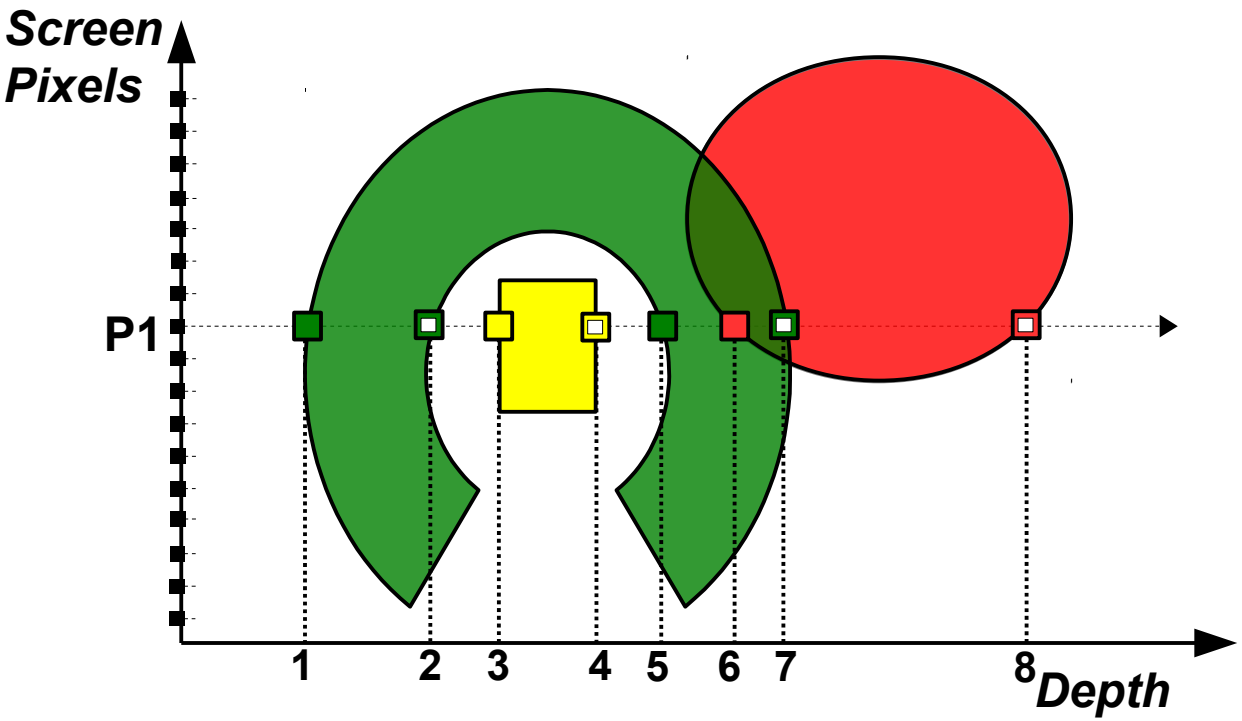
Depth-ranges

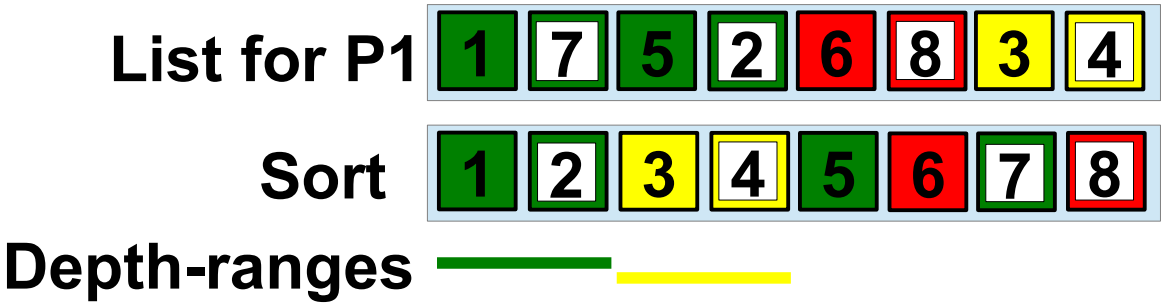
How does IBCD work?



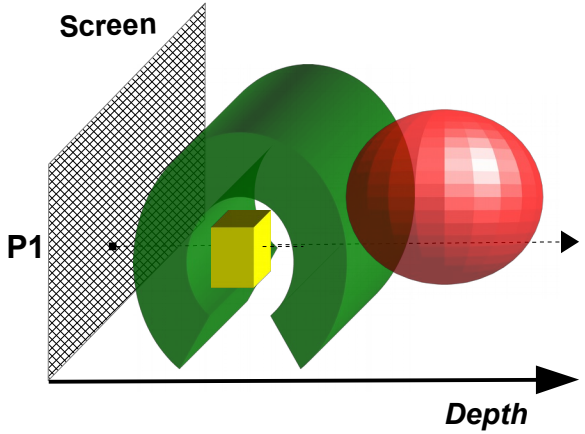
Steps of Image-Based CD

1) **Project** objects onto a plane (screen)
2) **Rasterize** surface of the objects and store depths in a **list**
3) **Sort** values by depth
4) **Detect** overlapping depth-ranges



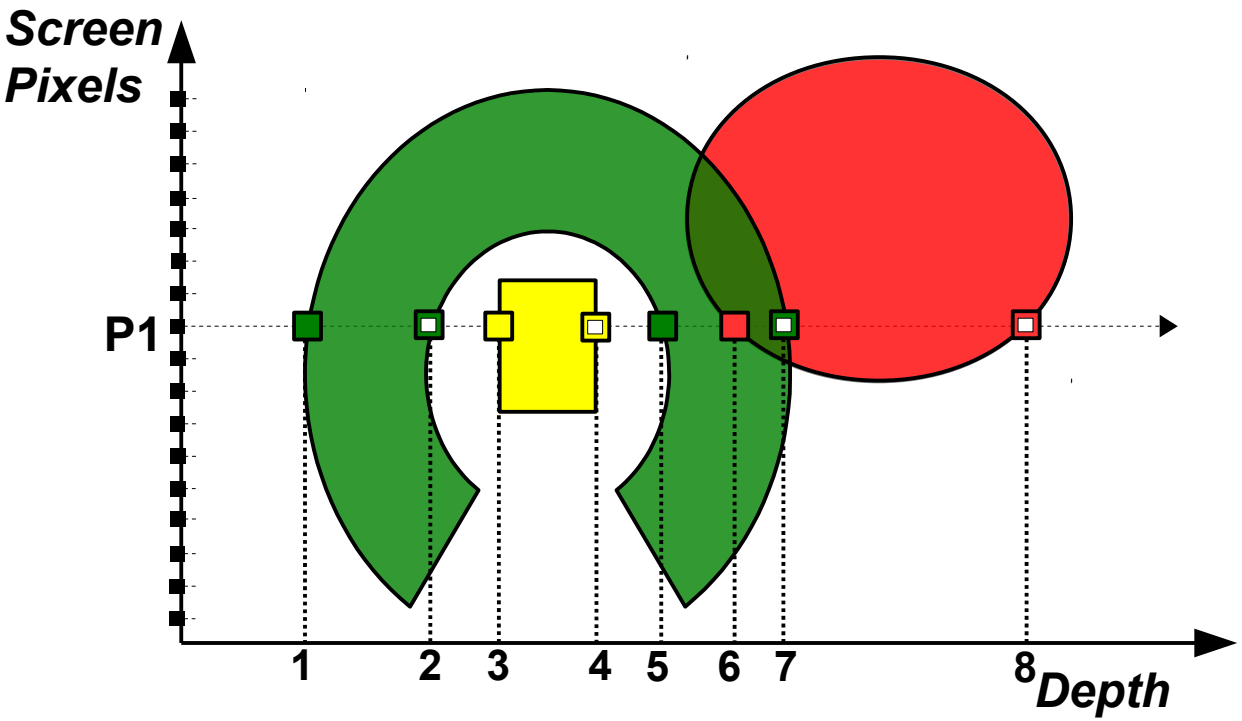


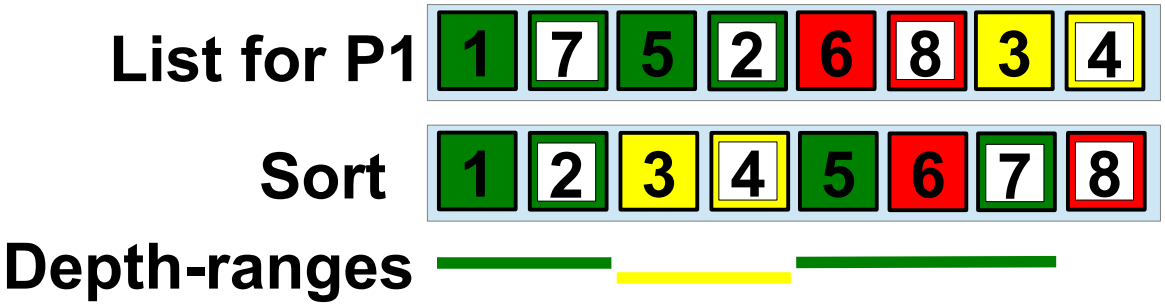
How does IBCD work?



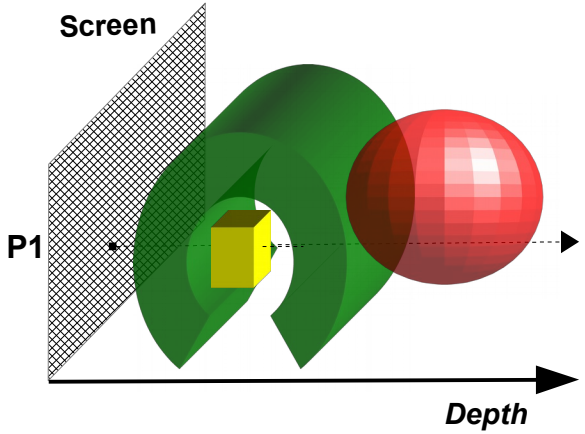
Steps of Image-Based CD

1) **Project** objects onto a plane (screen)
2) **Rasterize** surface of the objects and store depths in a **list**
3) **Sort** values by depth
4) **Detect** overlapping depth-ranges



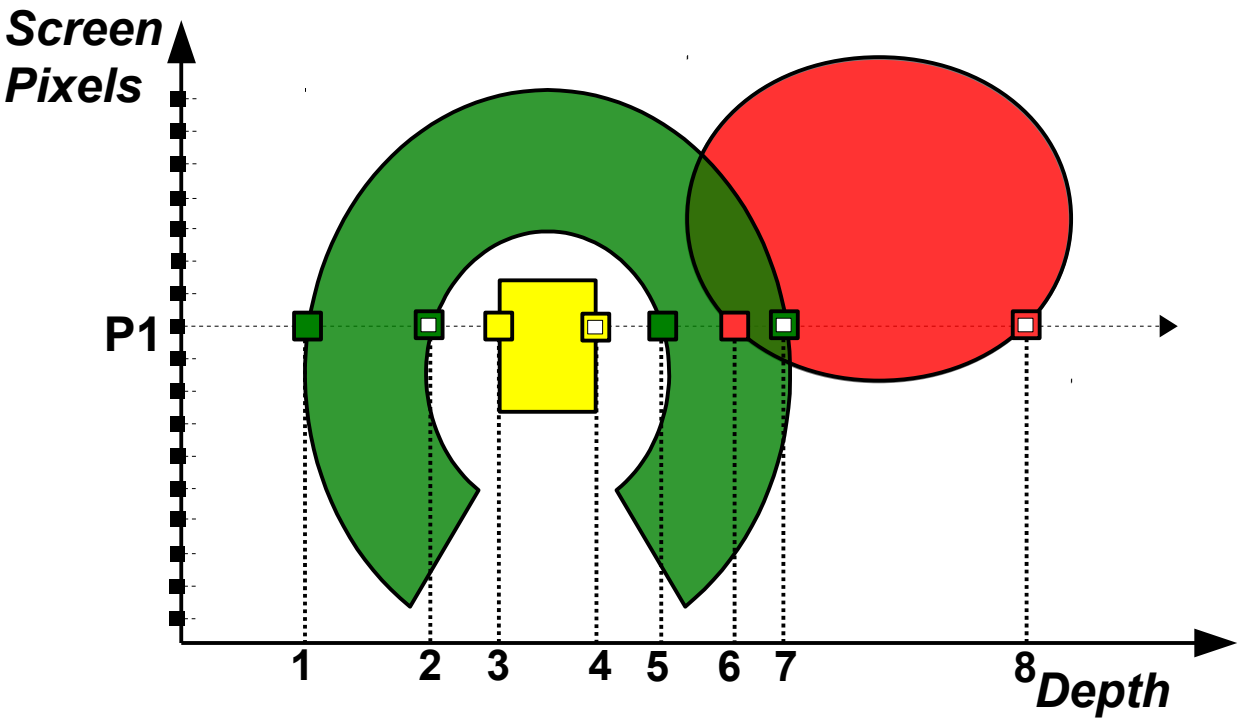


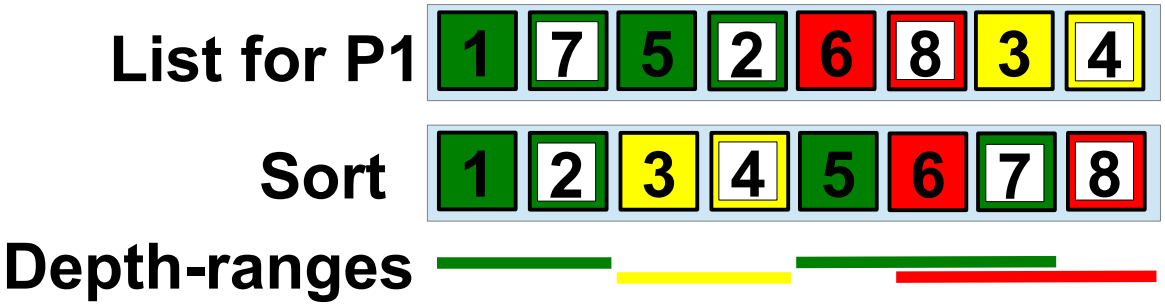
How does IBCD work?



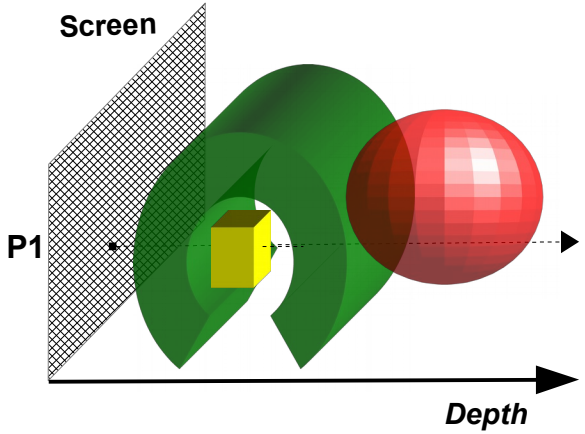
Steps of Image-Based CD

1) **Project** objects onto a plane (screen)
2) **Rasterize** surface of the objects and store depths in a **list**
3) **Sort** values by depth
4) **Detect** overlapping depth-ranges



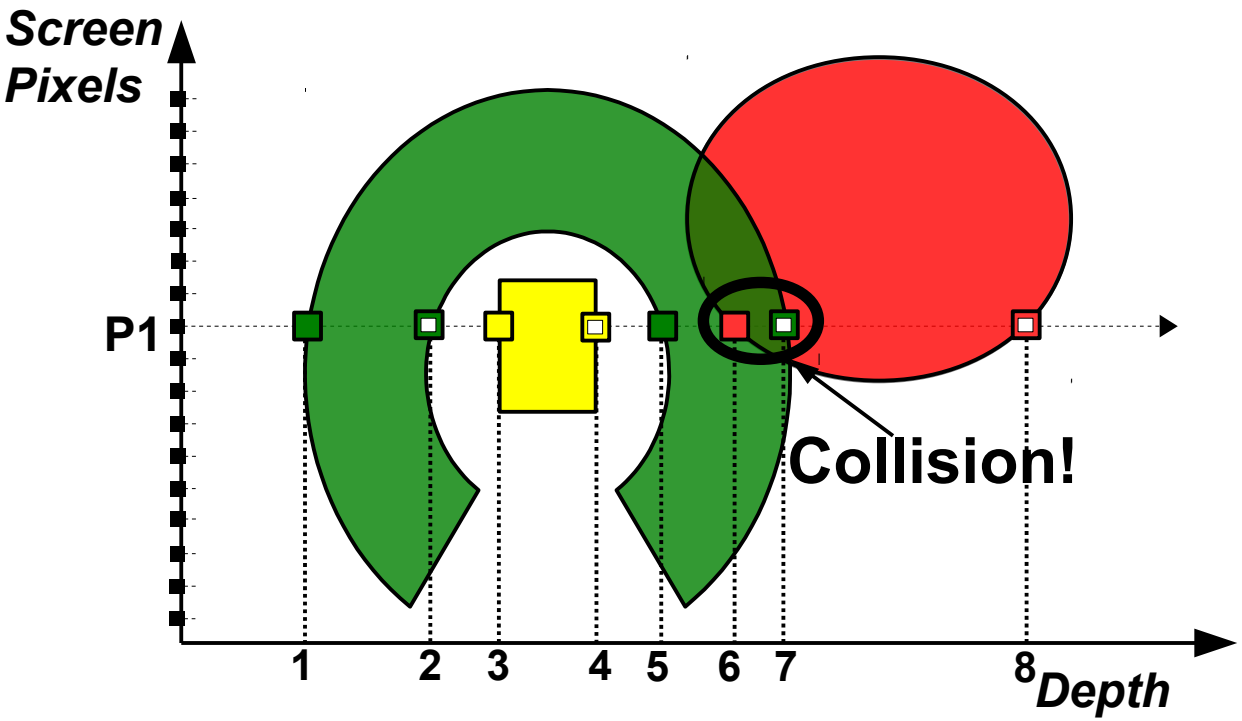


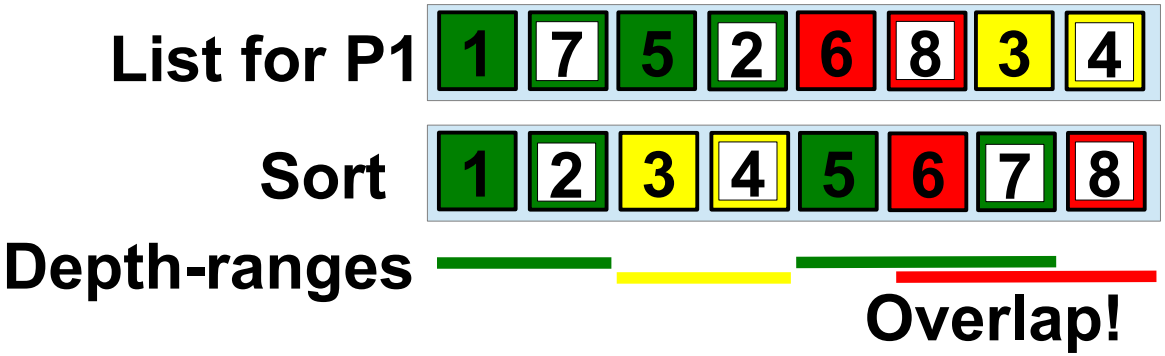
How does IBCD work?



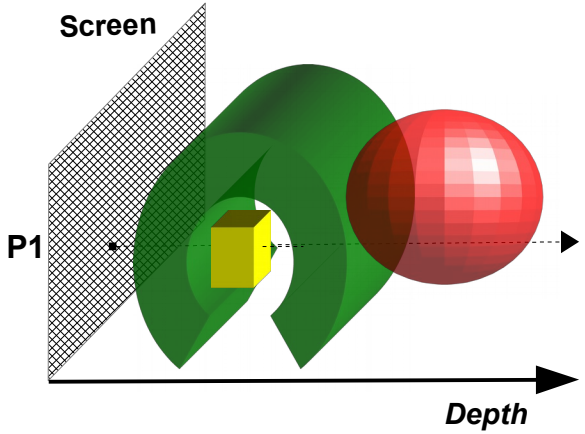
Steps of Image-Based CD

1) **Project** objects onto a plane (screen)
2) **Rasterize** surface of the objects and store depths in a **list**
3) **Sort** values by depth
4) **Detect** overlapping depth-ranges



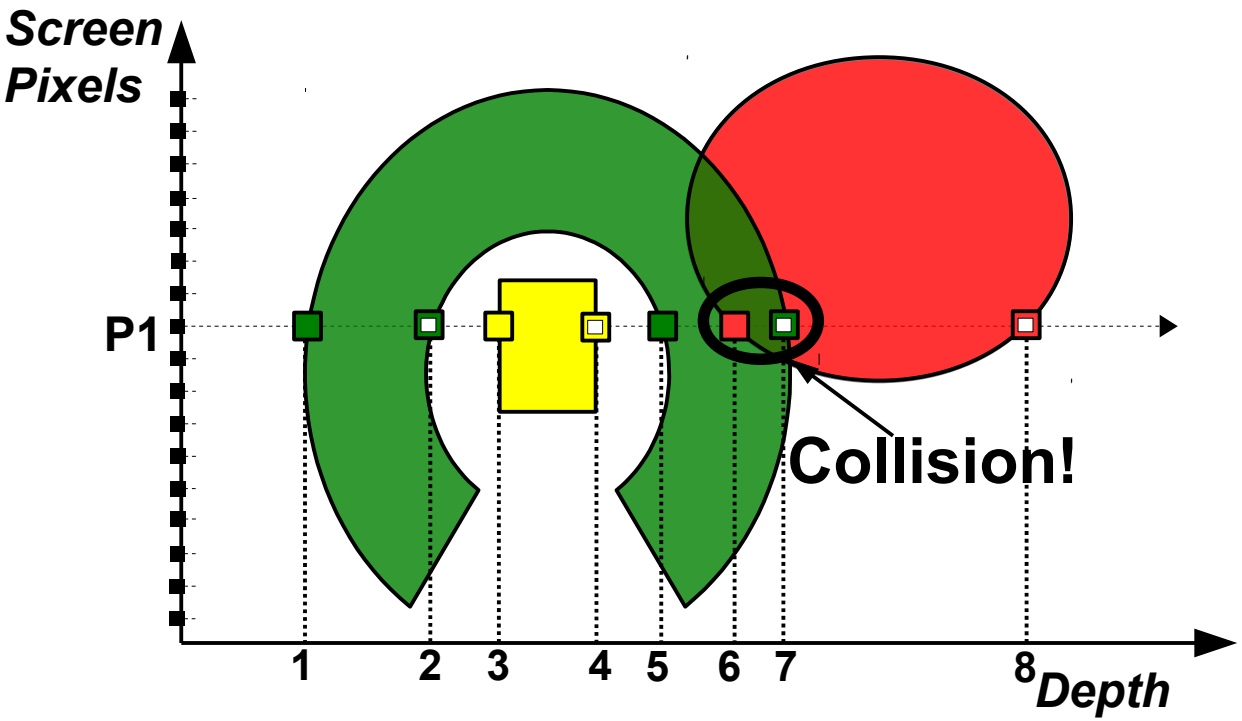


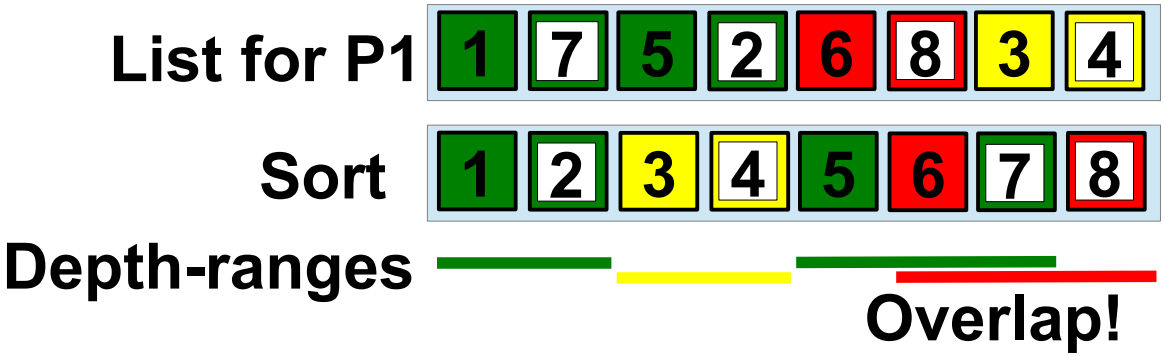
How does IBCD work?



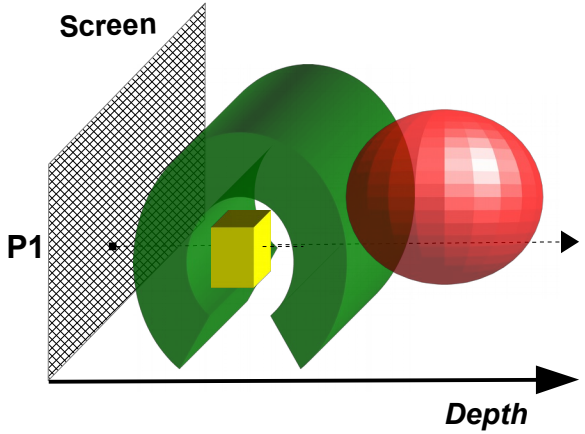
Steps of Image-Based CD

1) **Project** objects onto a plane (screen)
2) **Rasterize** surface of the objects and store depths in a **list**
3) **Sort** values by depth
4) **Detect** overlapping depth-ranges



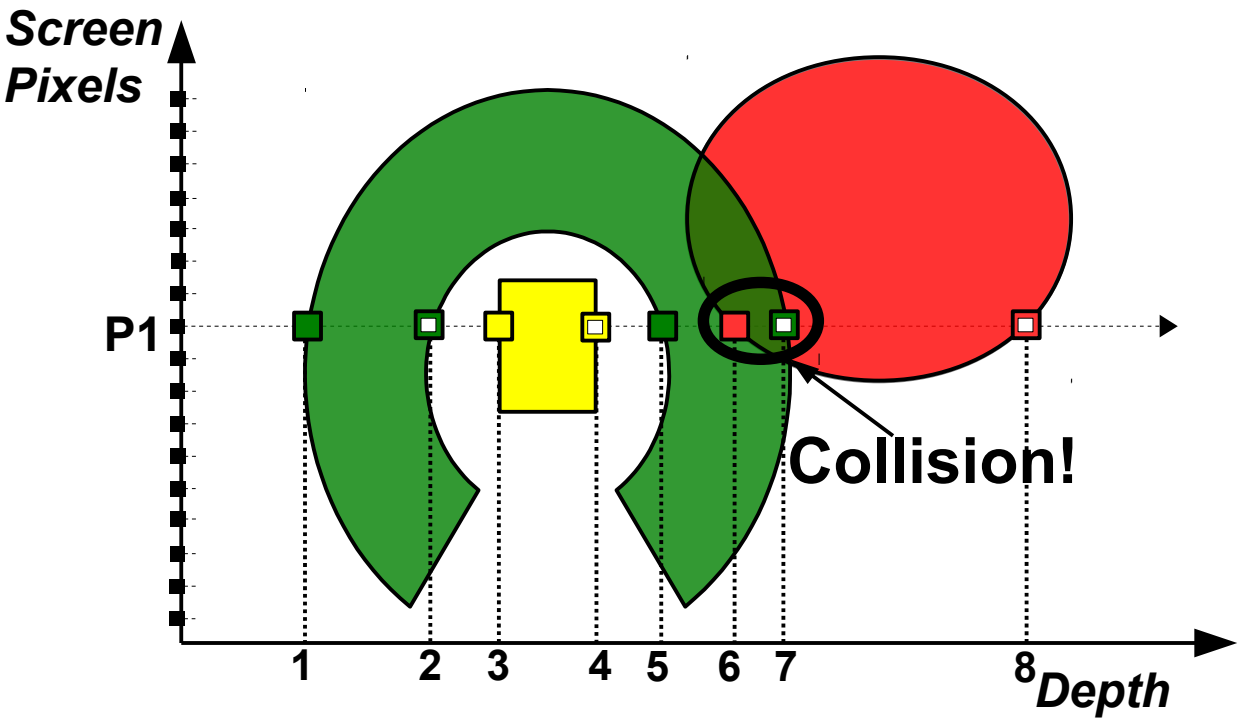


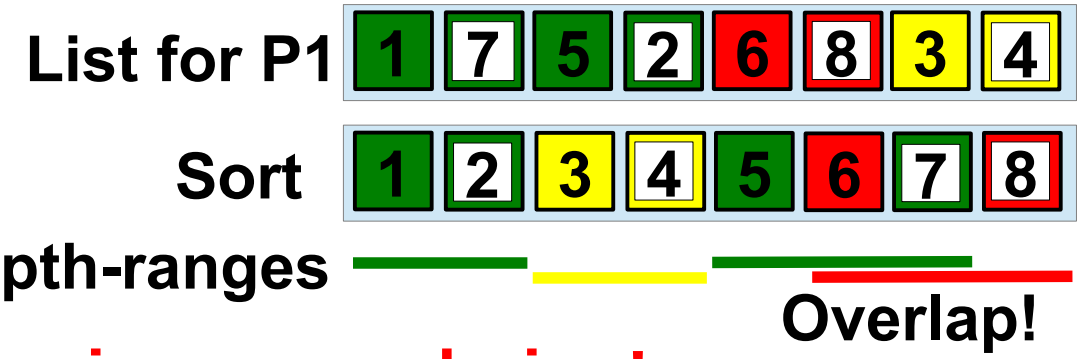
How does IBCD work?



Steps of Image-Based CD

1) **Project** objects onto a plane (screen)
2) **Rasterize** surface of the objects and store depths in a **list**
3) **Sort** values by depth
4) **Detect** overlapping depth-ranges





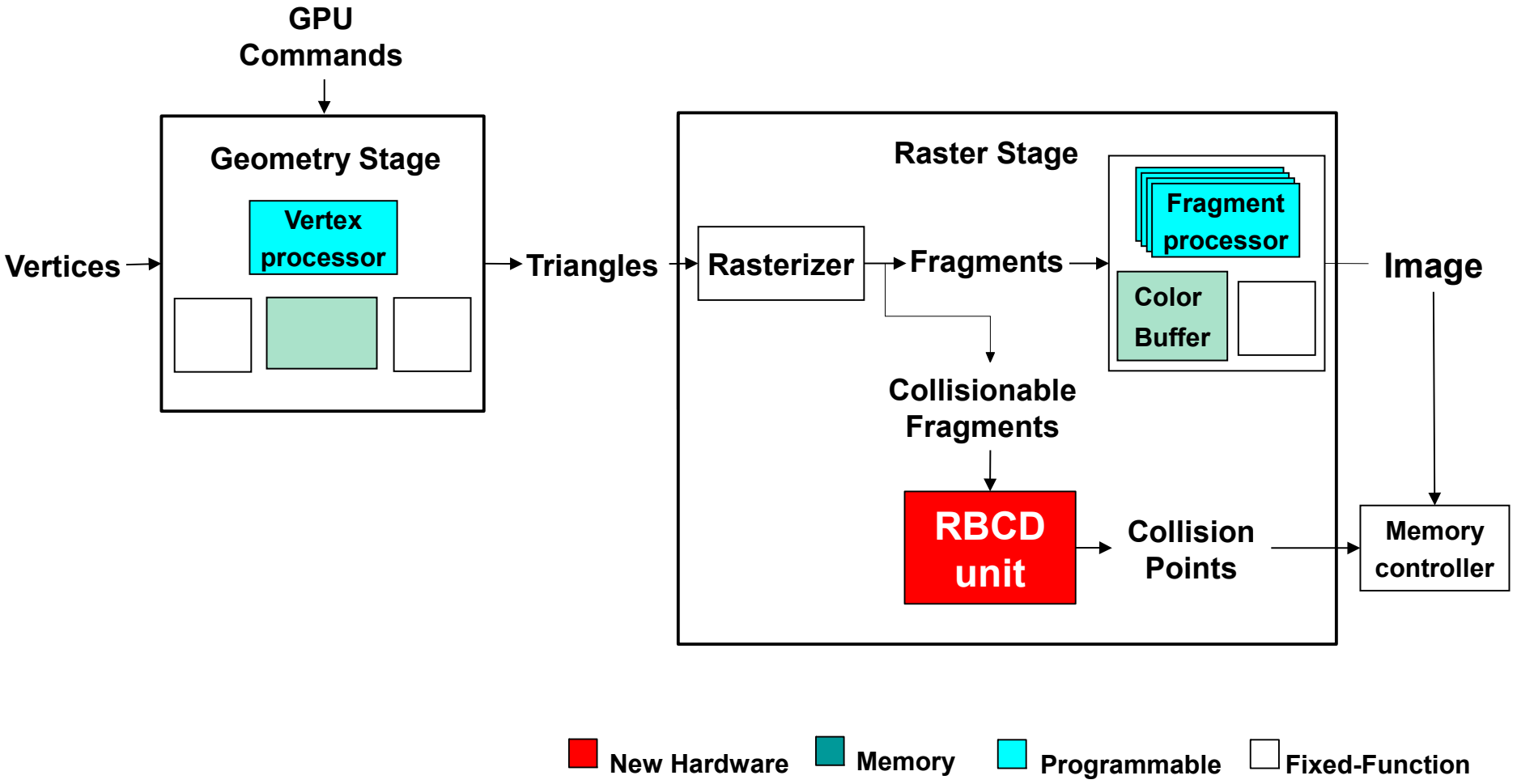
Already done for image rendering!

Outline

1. Motivation
2. Collision Detection (CD)
3. **Render-Based CD in the GPU**
4. Results
5. Conclusion

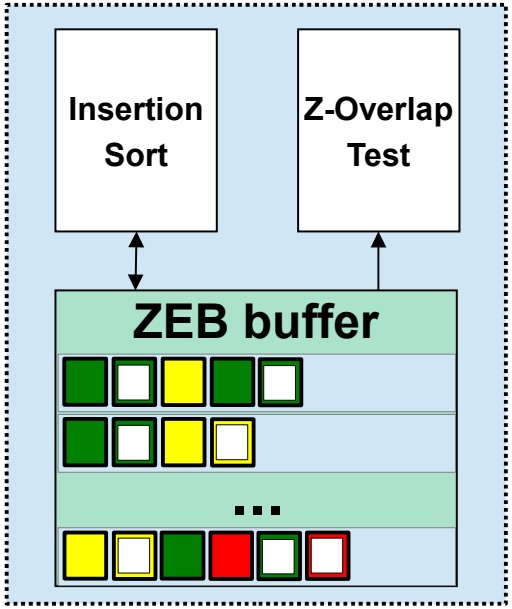
CD Integration into GPU pipeline

Graphics Pipeline from 10,000 feet



RBCD unit

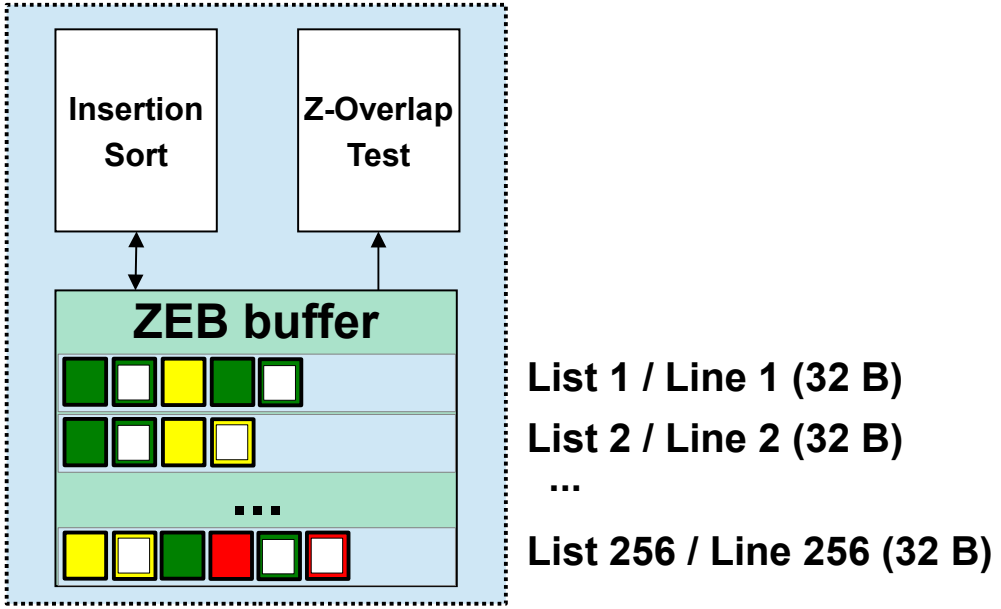
1) Store fragments sorted by depth
2) Detects collision points



List 1 / Line 1 (32 B)
List 2 / Line 2 (32 B)
...
List 256 / Line 256 (32 B)

RBCD unit

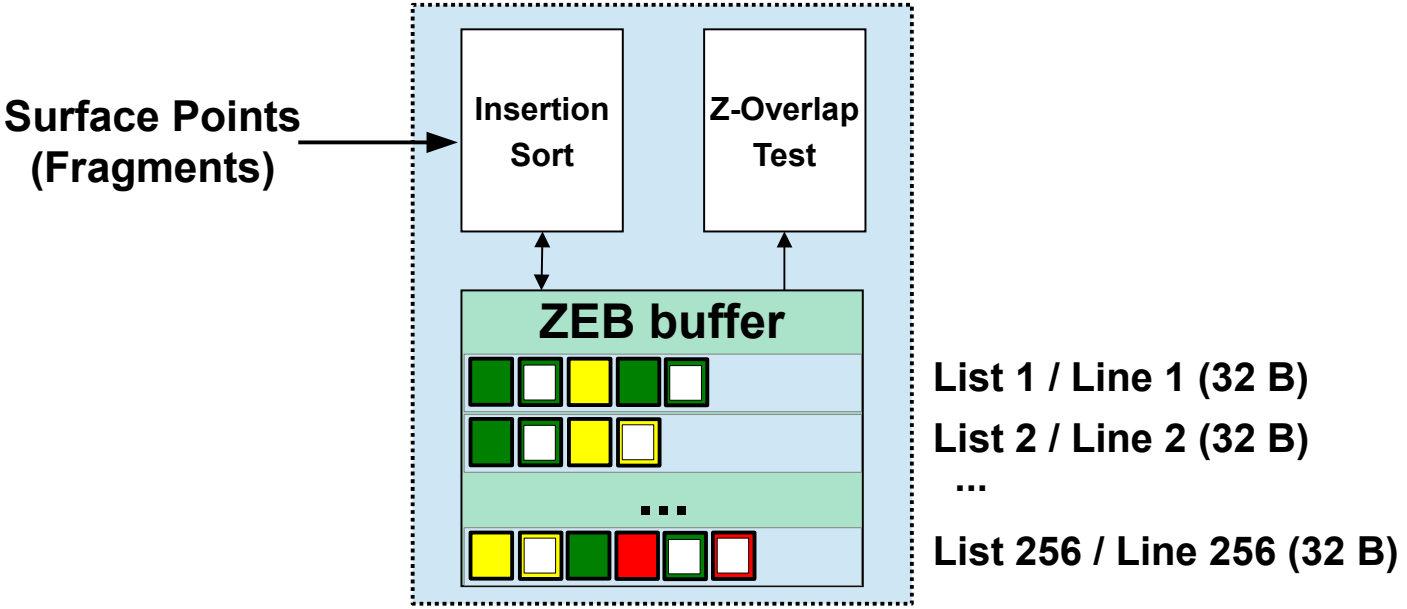
1) Store fragments sorted by depth
2) Detects collision points



ZEB buffer (on-chip array)
1 list of fragments per every pixel
16x16 lists, 8 entries per list
8 KB (16x16x8x32B)

RBCD unit

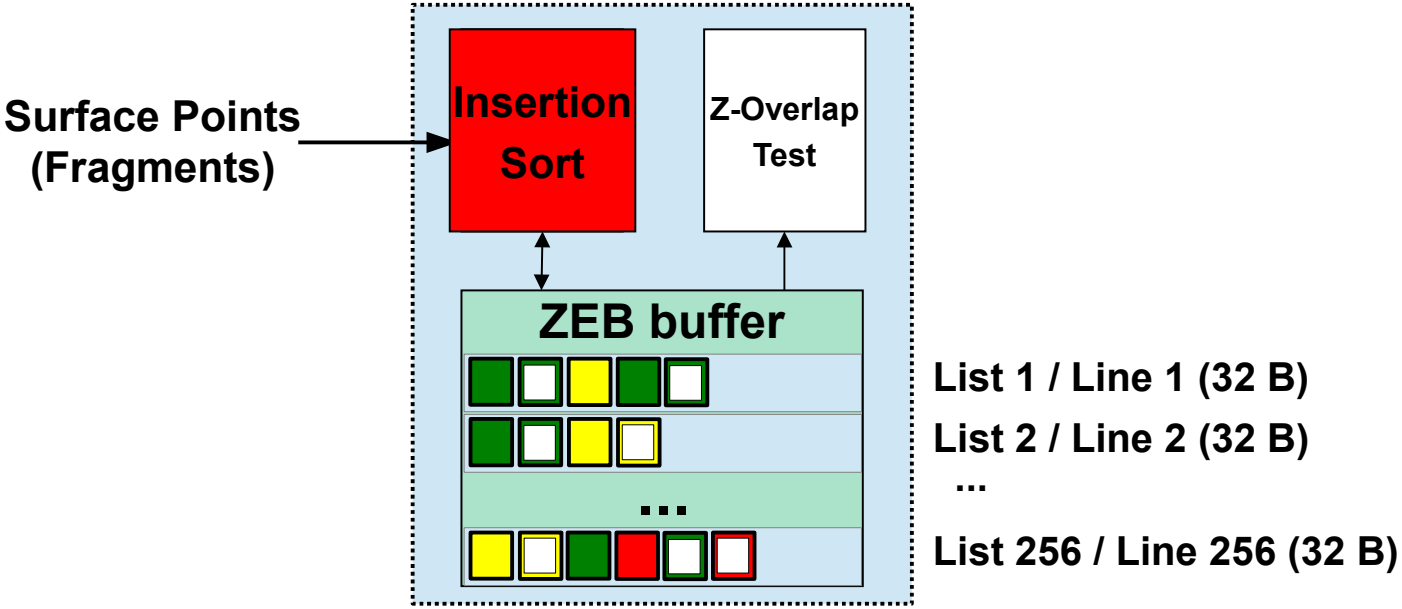
1) Store fragments sorted by depth
2) Detects collision points



ZEB buffer (on-chip array)
1 list of fragments per every pixel
16x16 lists, 8 entries per list
8 KB (16x16x8x32B)

RBCD unit

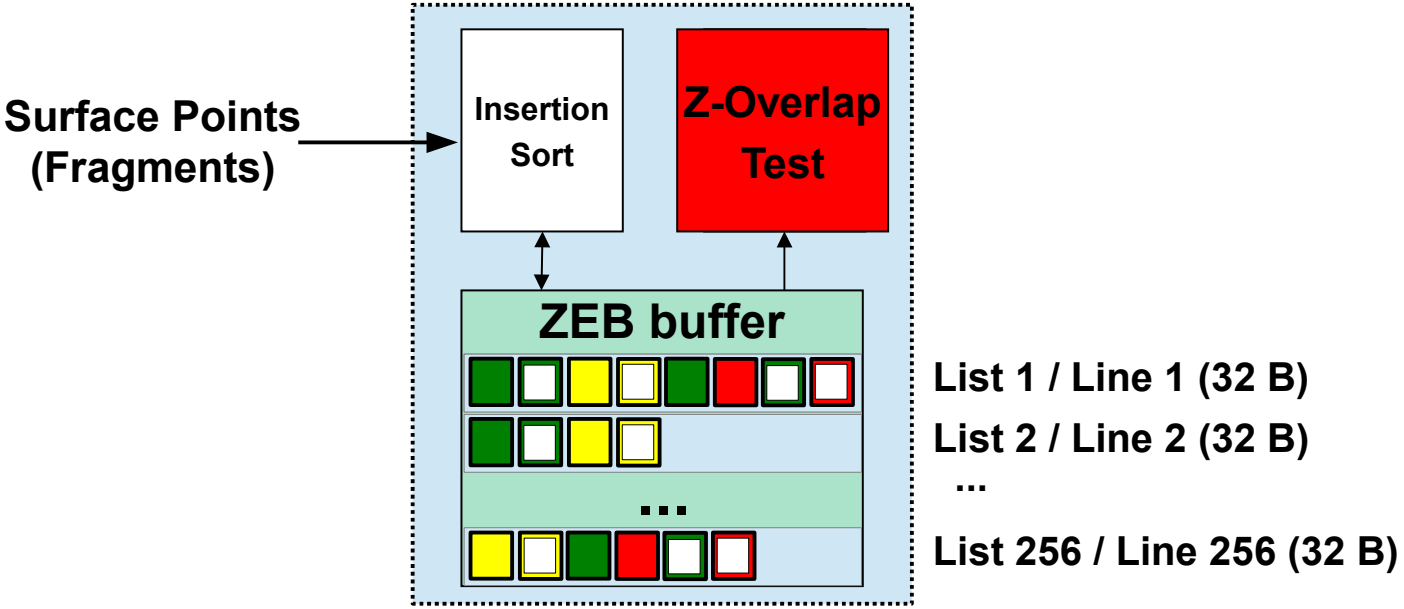
1) Store fragments sorted by depth
2) Detects collision points



ZEB buffer (on-chip array)
1 list of fragments per every pixel
16x16 lists, 8 entries per list
8 KB (16x16x8x32B)

RBCD unit

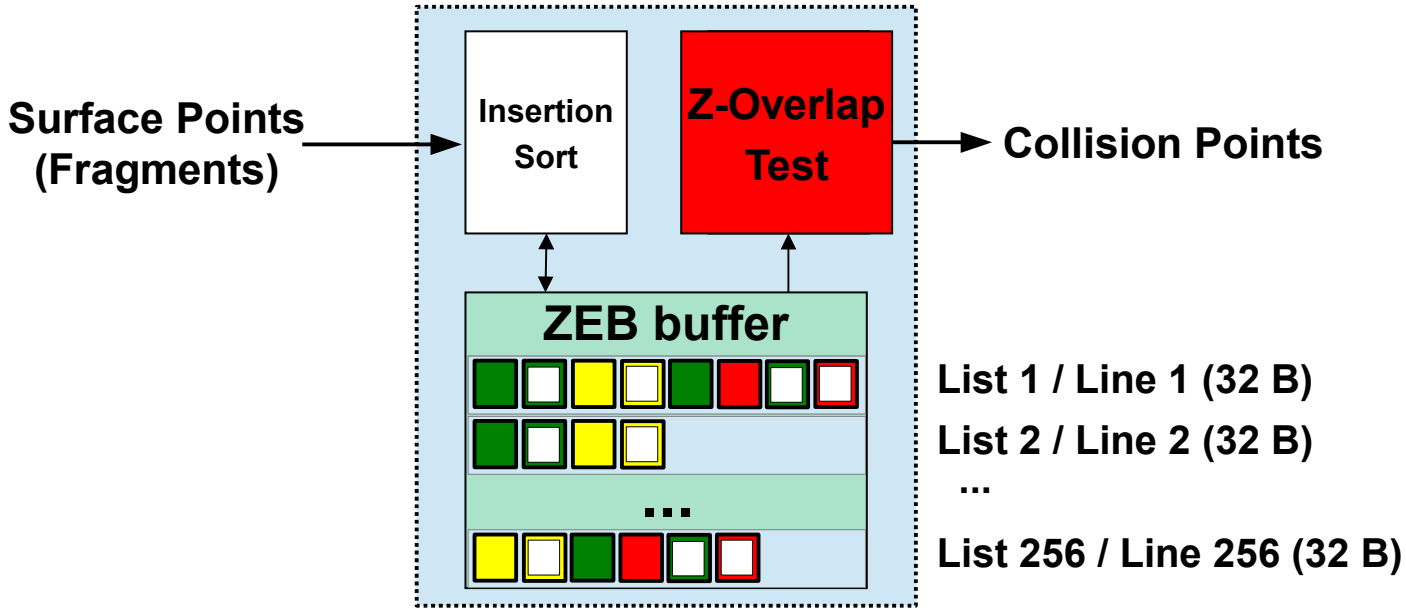
1) Store fragments sorted by depth
2) Detects collision points



ZEB buffer (on-chip array)
1 list of fragments per every pixel
16x16 lists, 8 entries per list
8 KB (16x16x8x32B)

RBCD unit

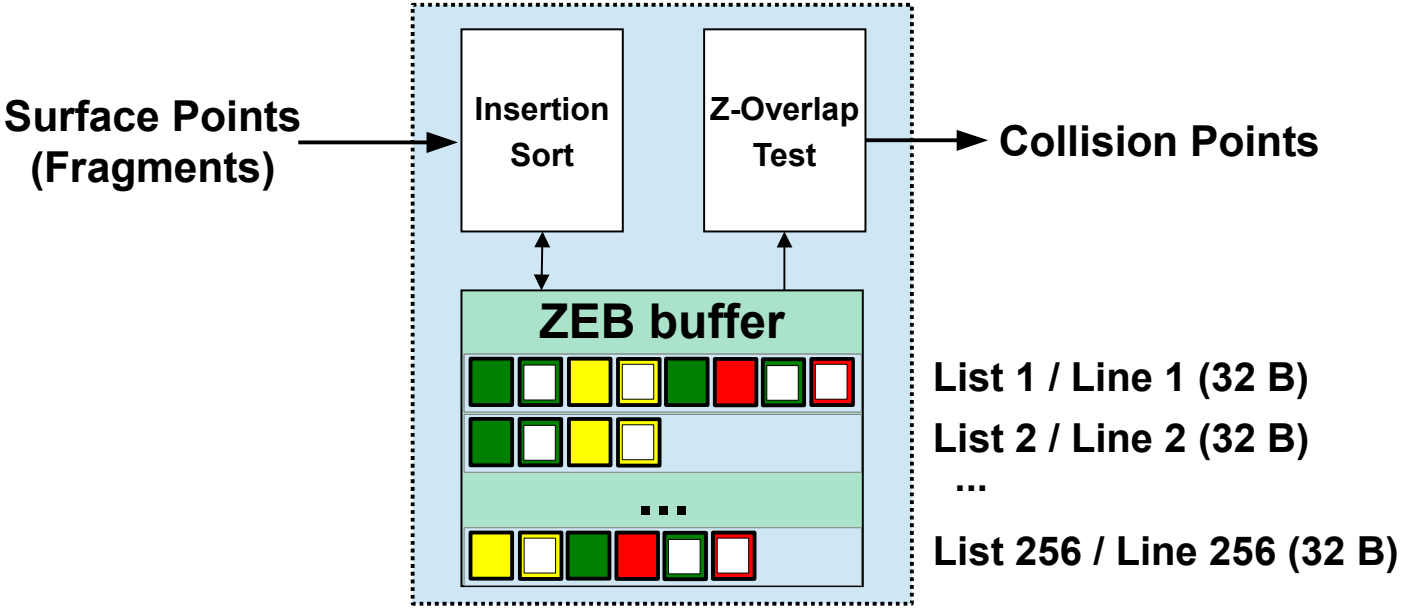
1) Store fragments sorted by depth
2) Detects collision points



ZEB buffer (on-chip array)
1 list of fragments per every pixel
16x16 lists, 8 entries per list
8 KB (16x16x8x32B)

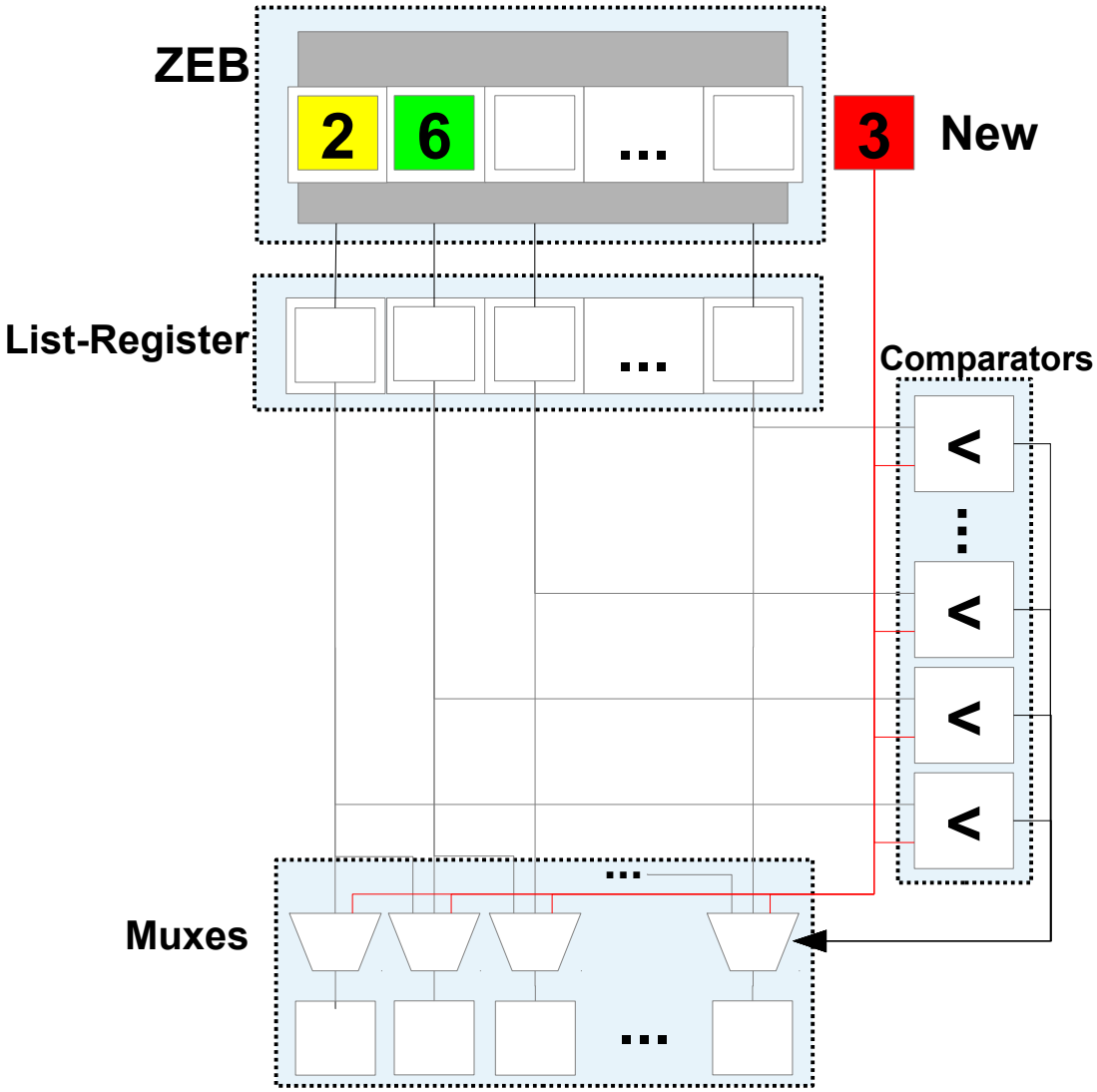
RBCD unit

1) Store fragments sorted by depth
2) Detects collision points



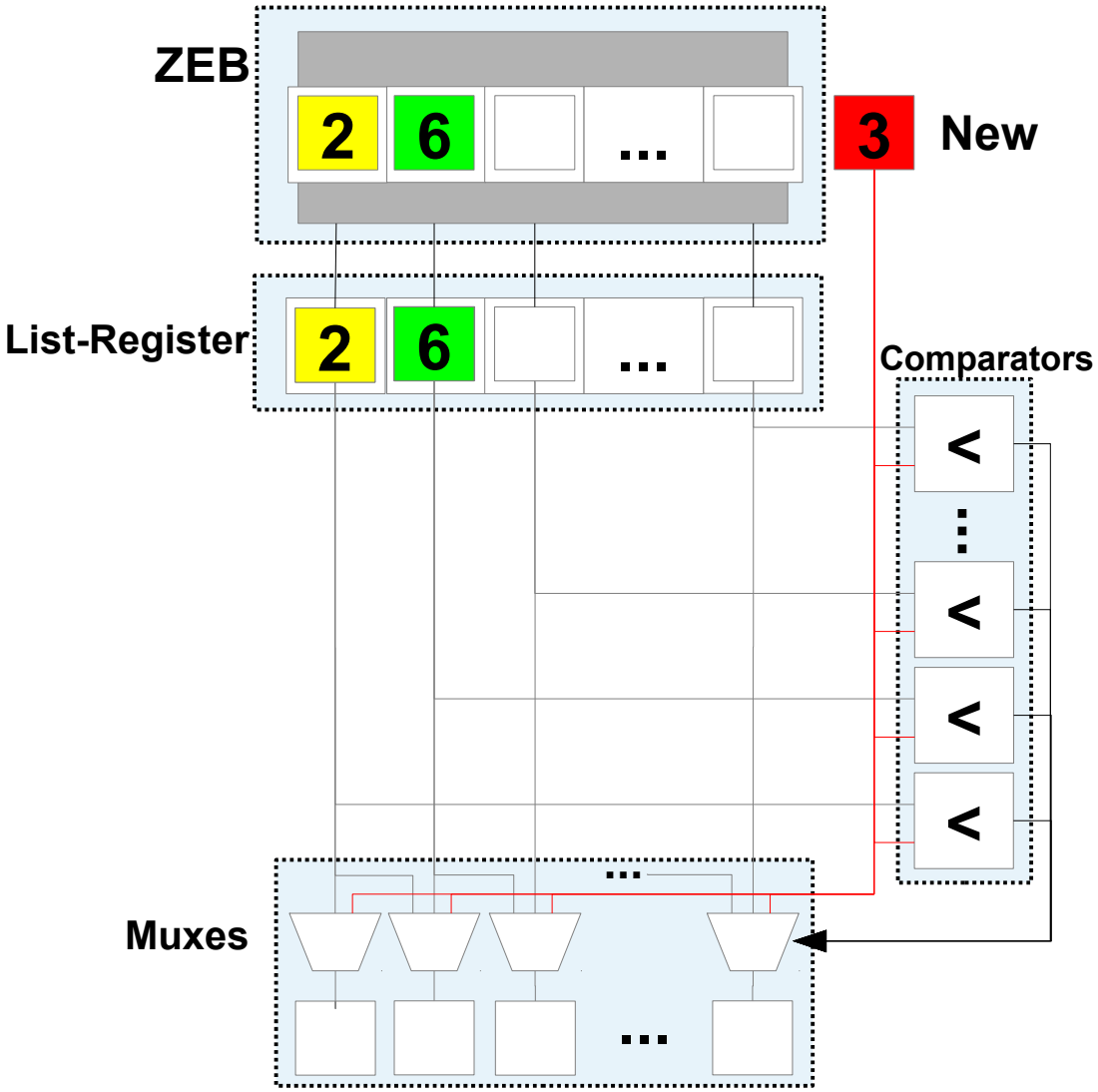
ZEB buffer (on-chip array)
1 list of fragments per every pixel
16x16 lists, 8 entries per list
8 KB (16x16x8x32B)

Insertion Sort



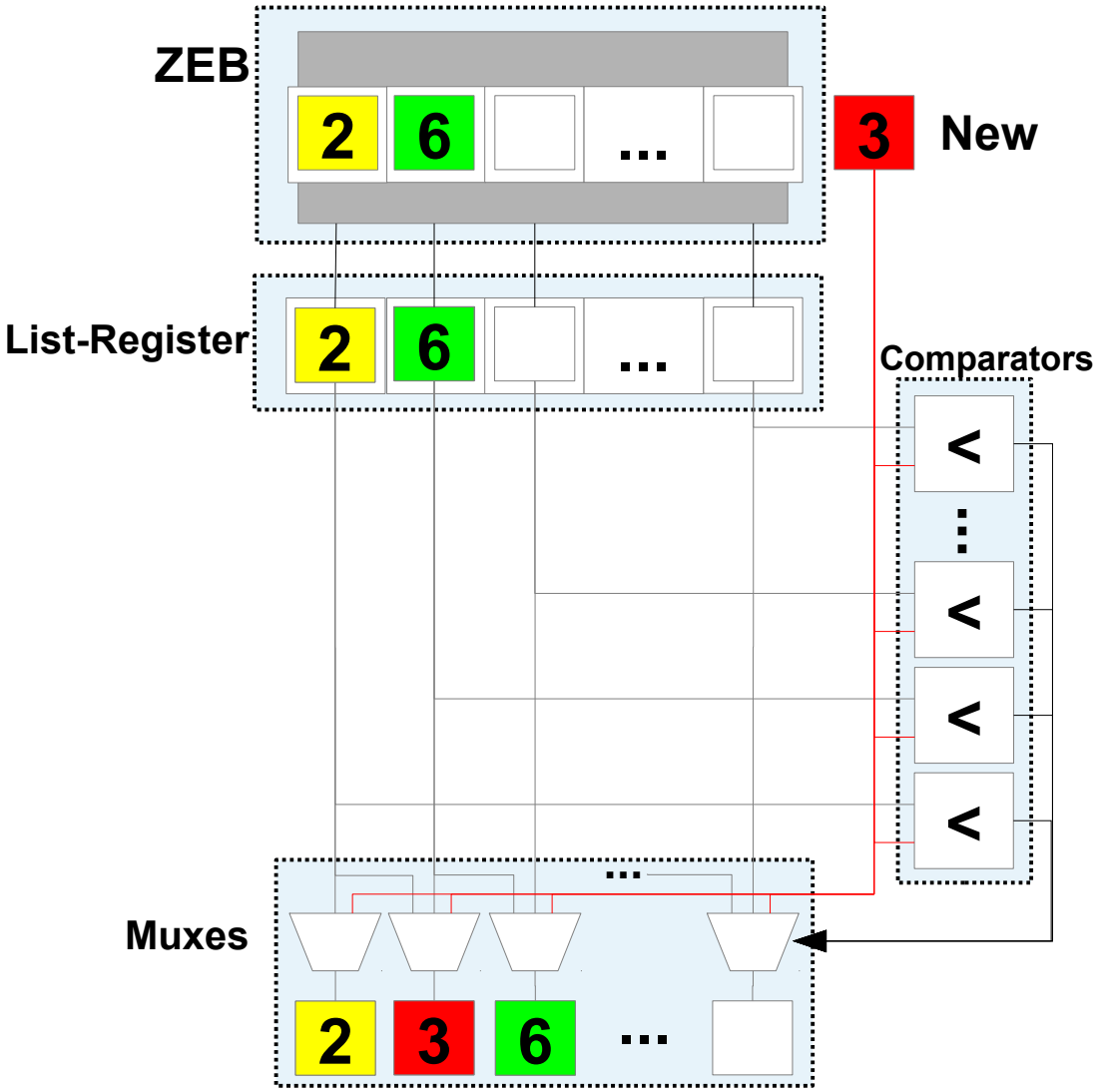
Read one list

Insertion Sort



Read one list

Insertion Sort

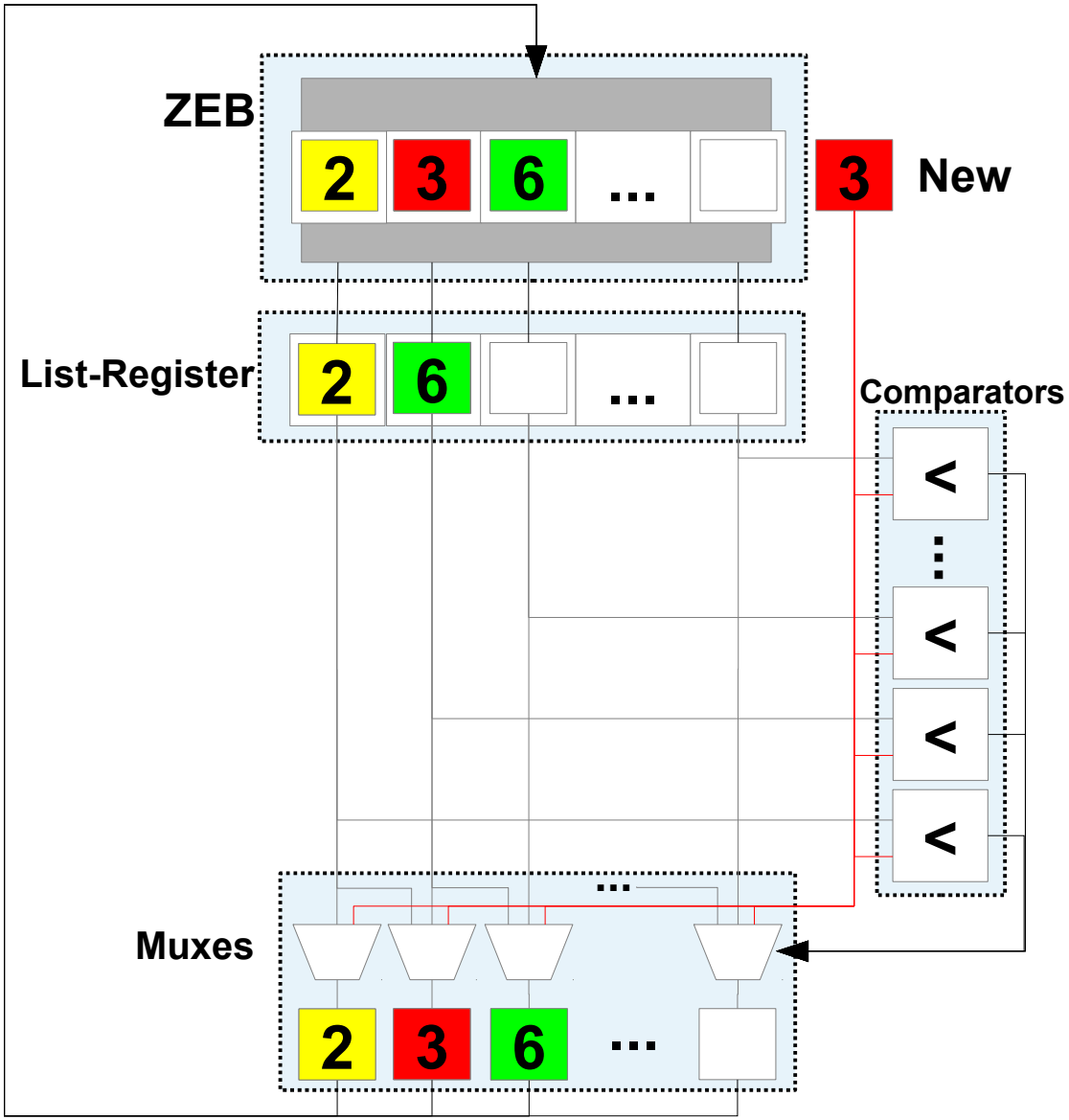


Read one list

Compare

Shift

Insertion Sort

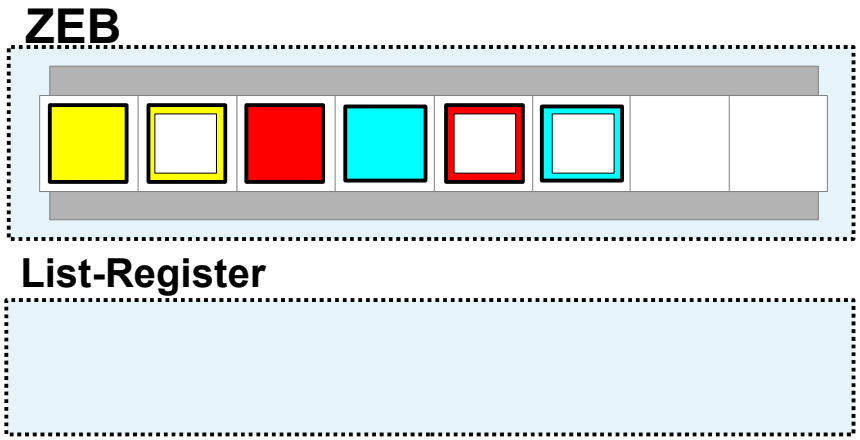


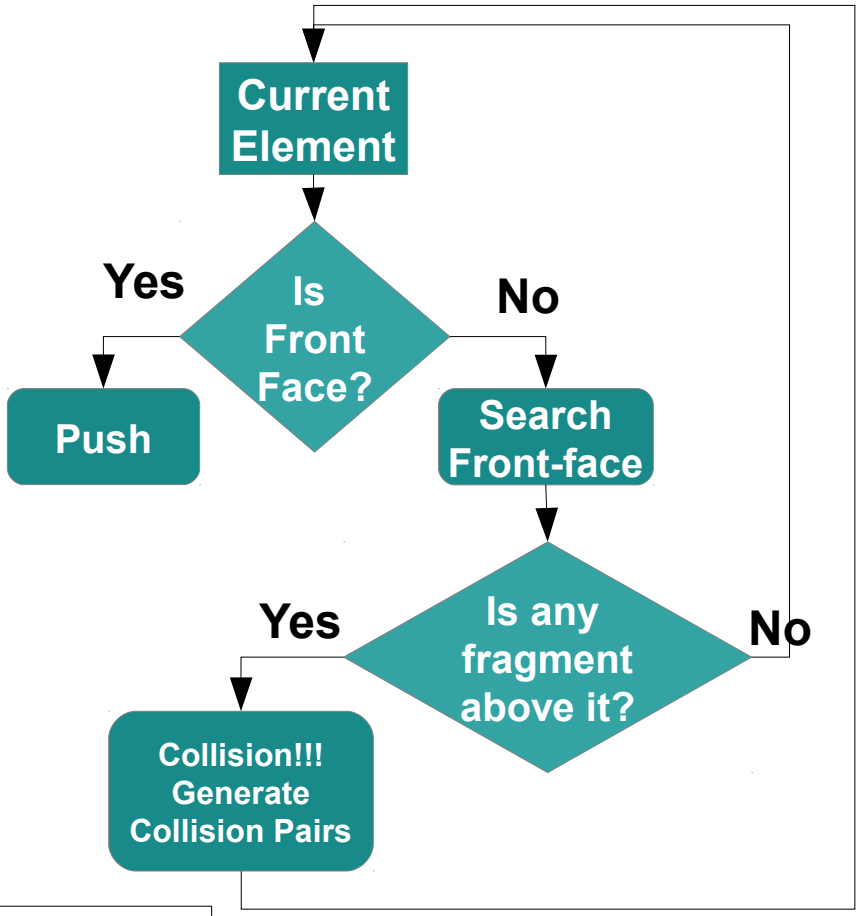
Read one list

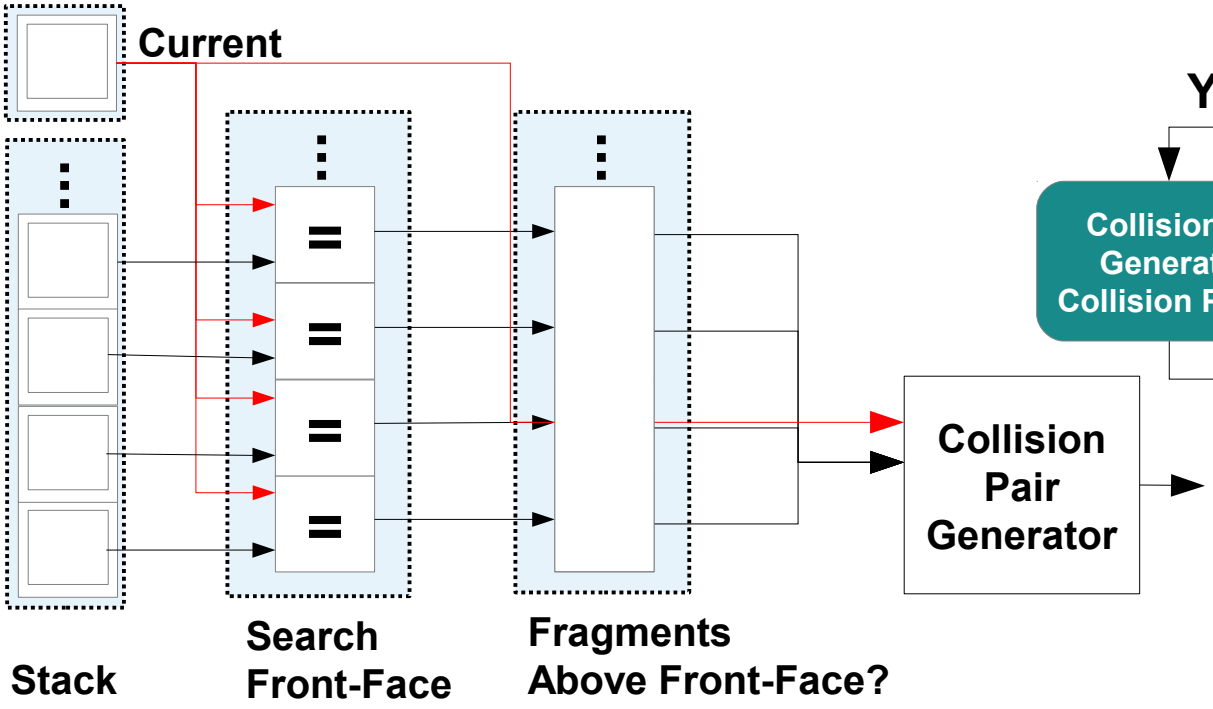
Compare
Shift

Write new list

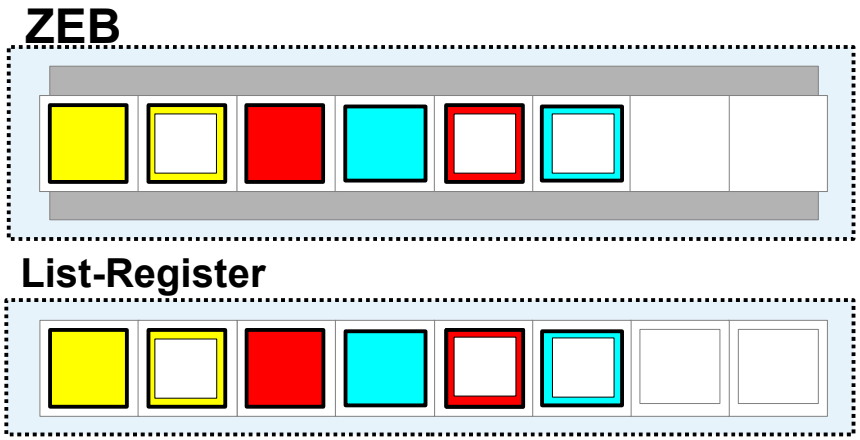
Z-Overlap Test

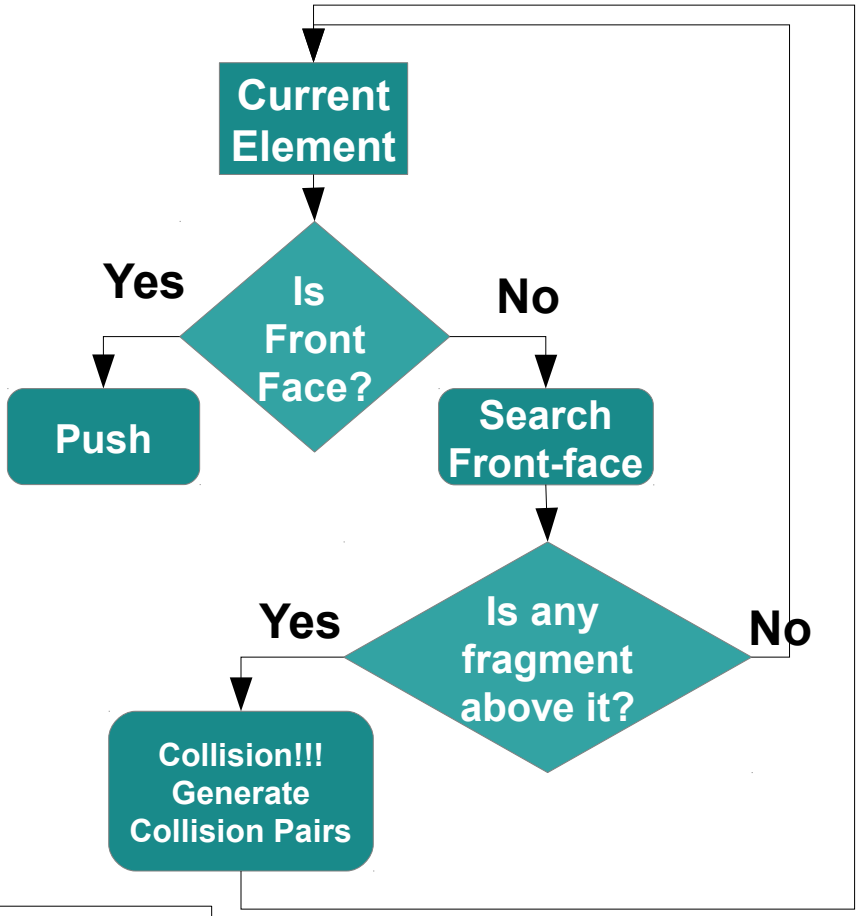


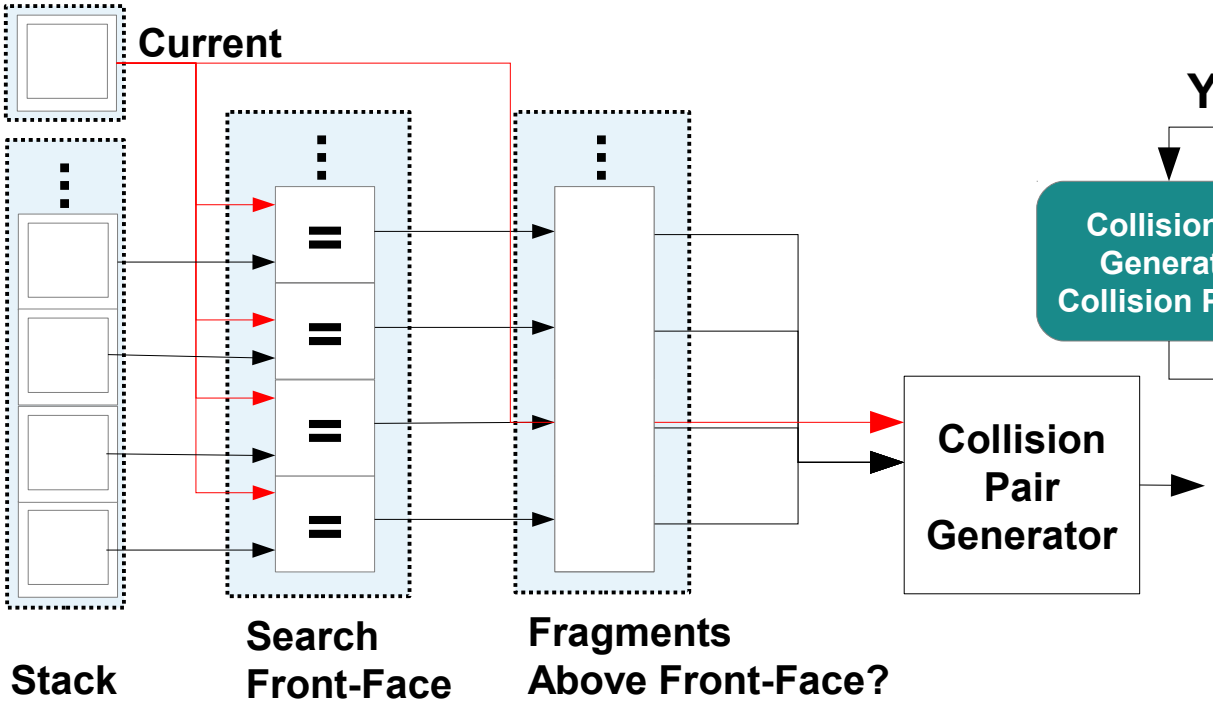




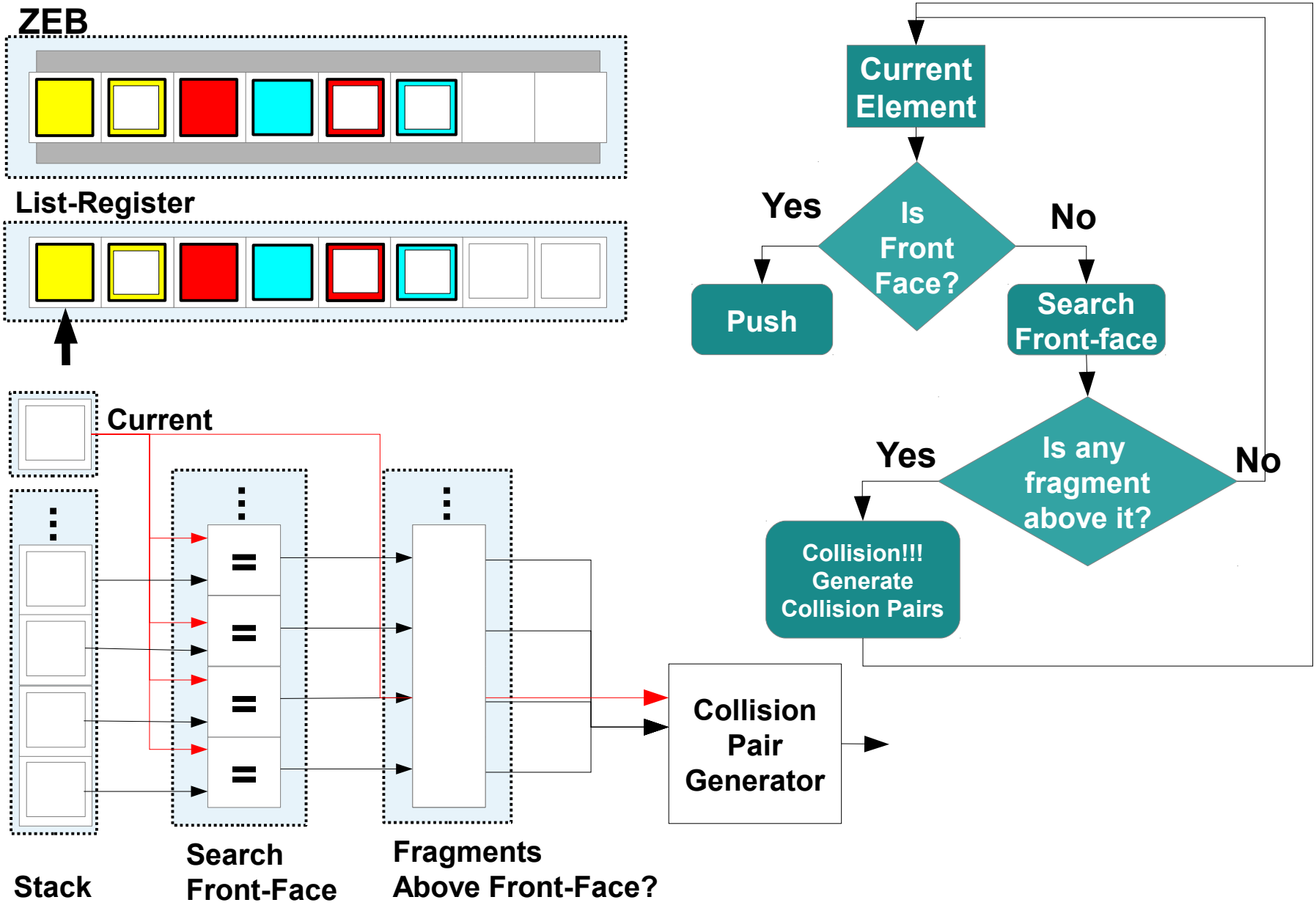
Z-Overlap Test



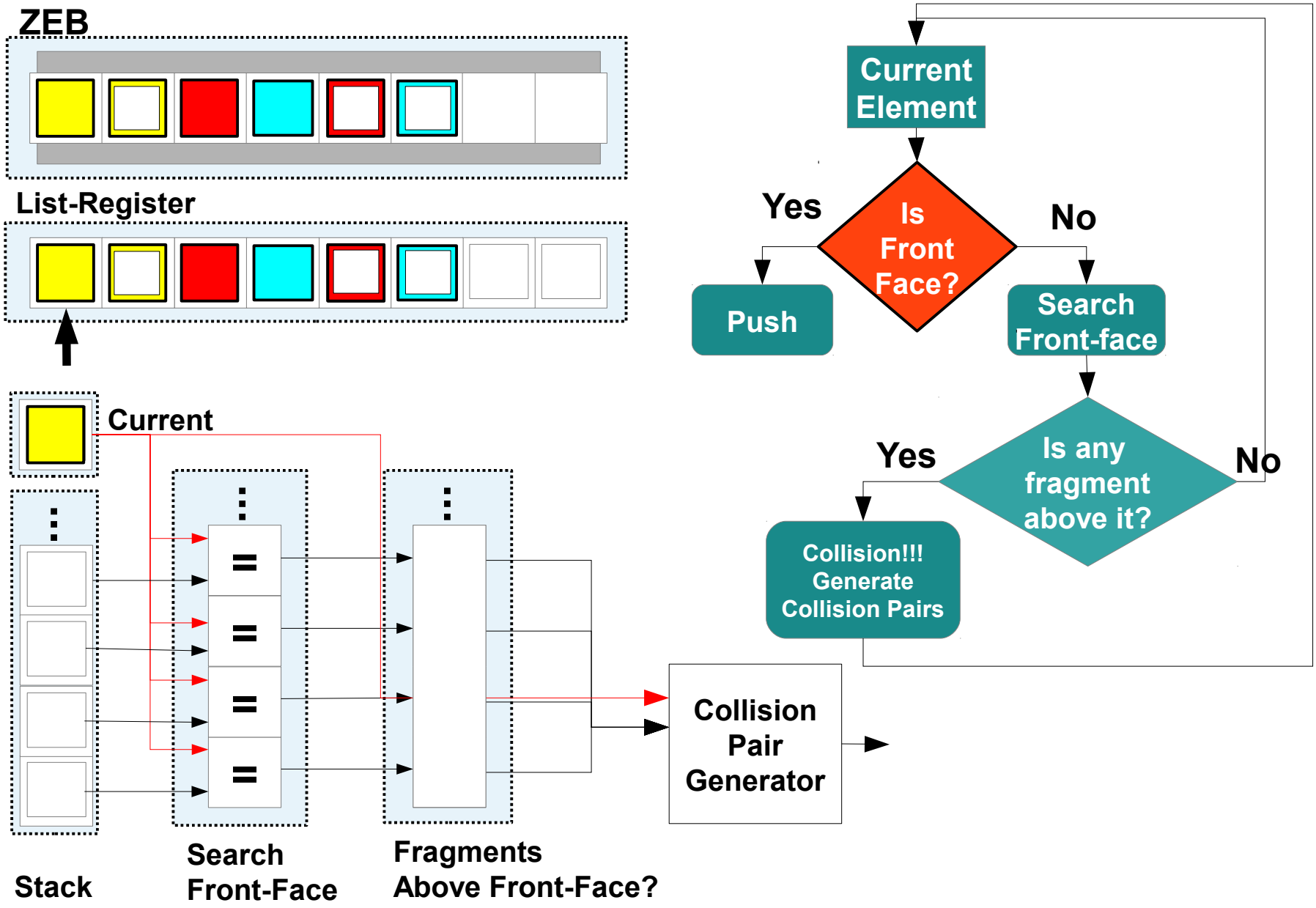




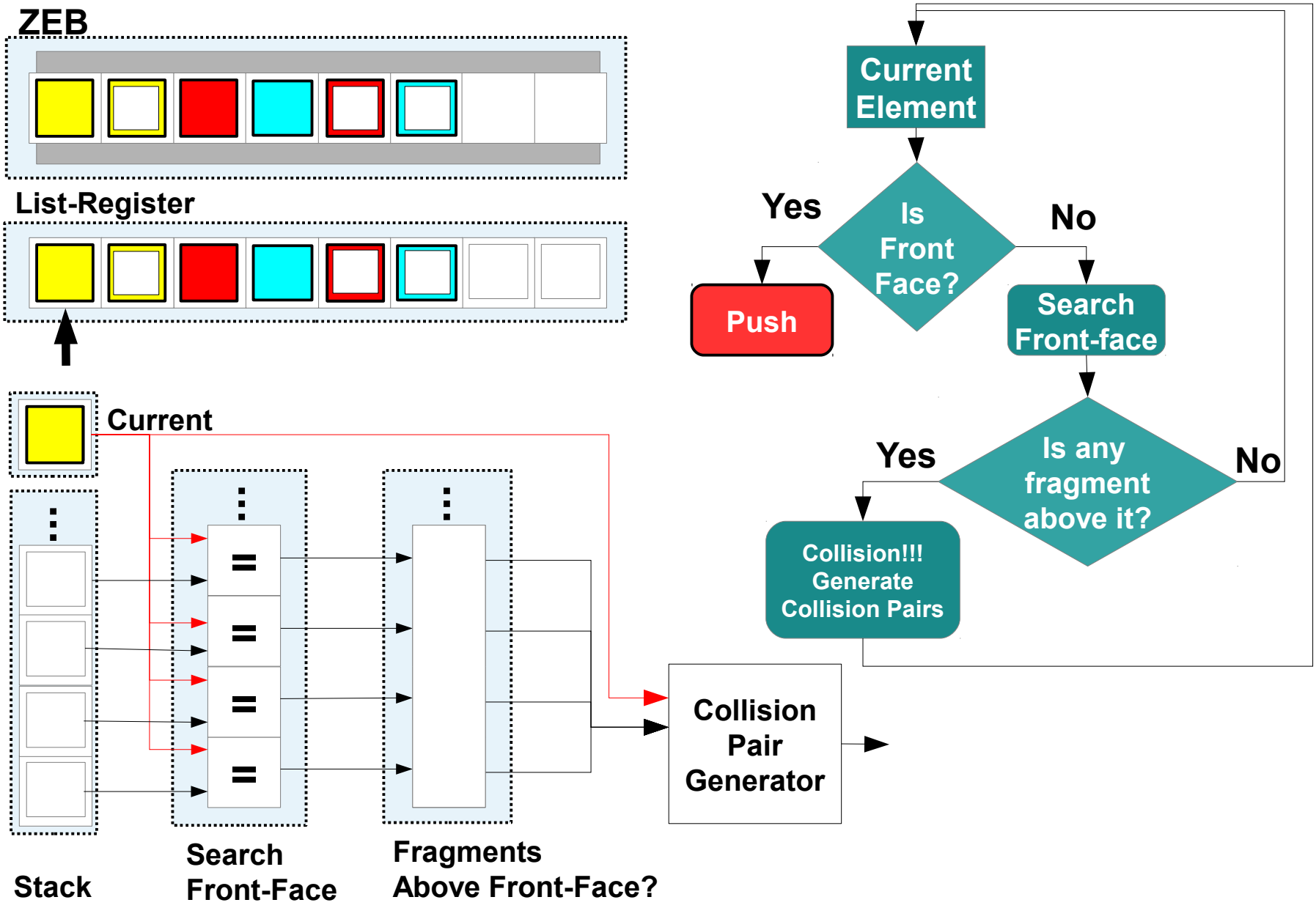
Z-Overlap Test



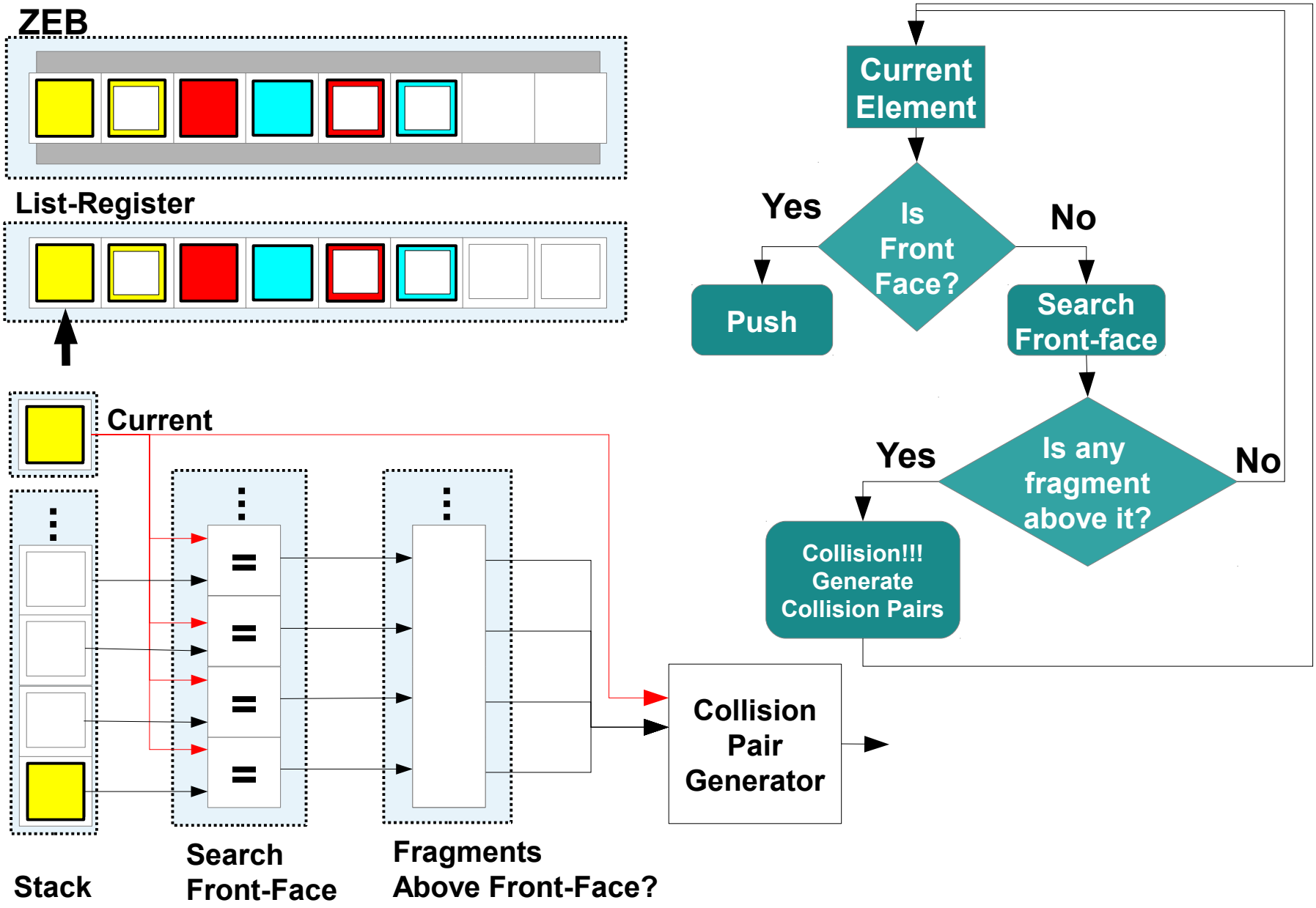
Z-Overlap Test



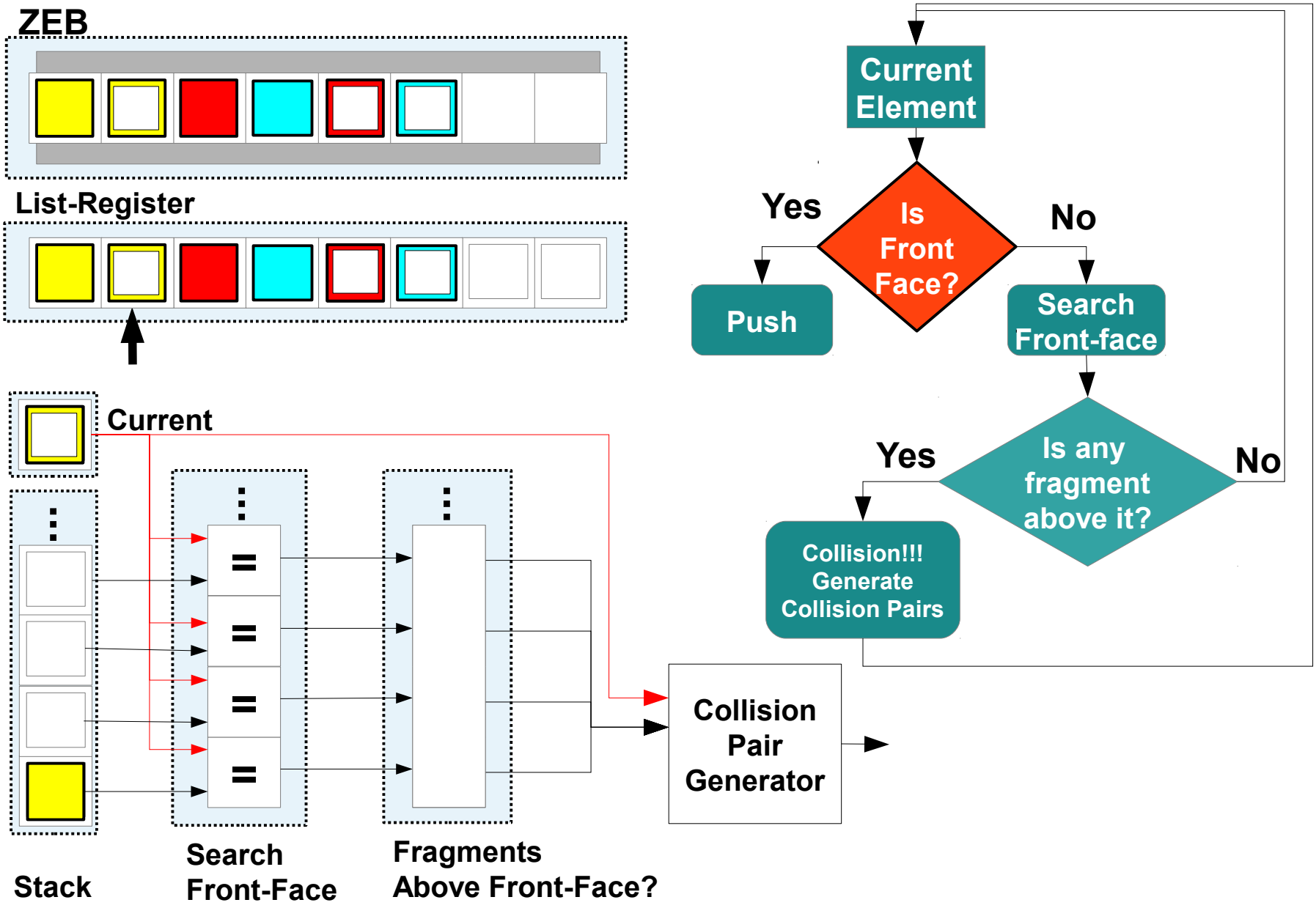
Z-Overlap Test



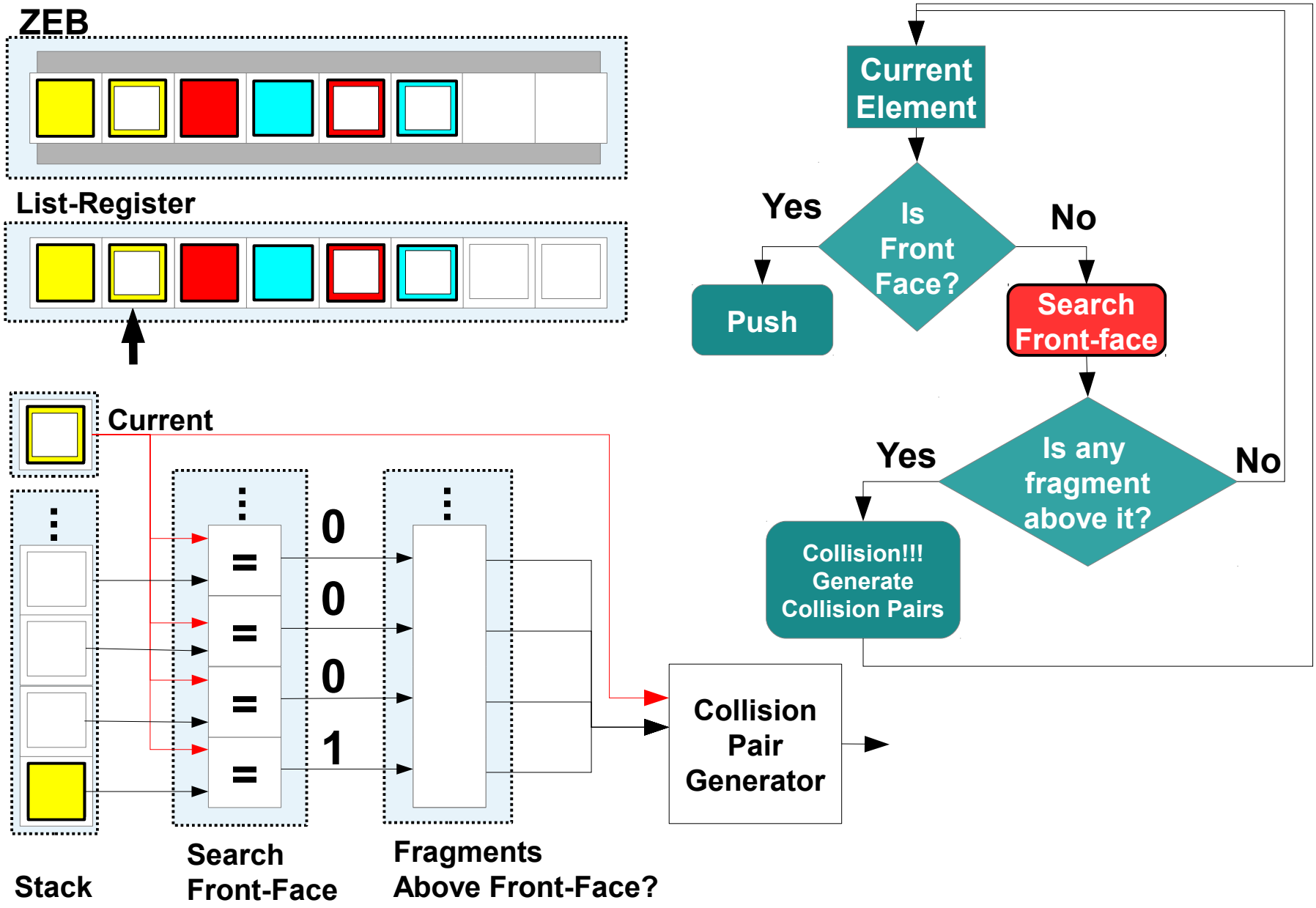
Z-Overlap Test



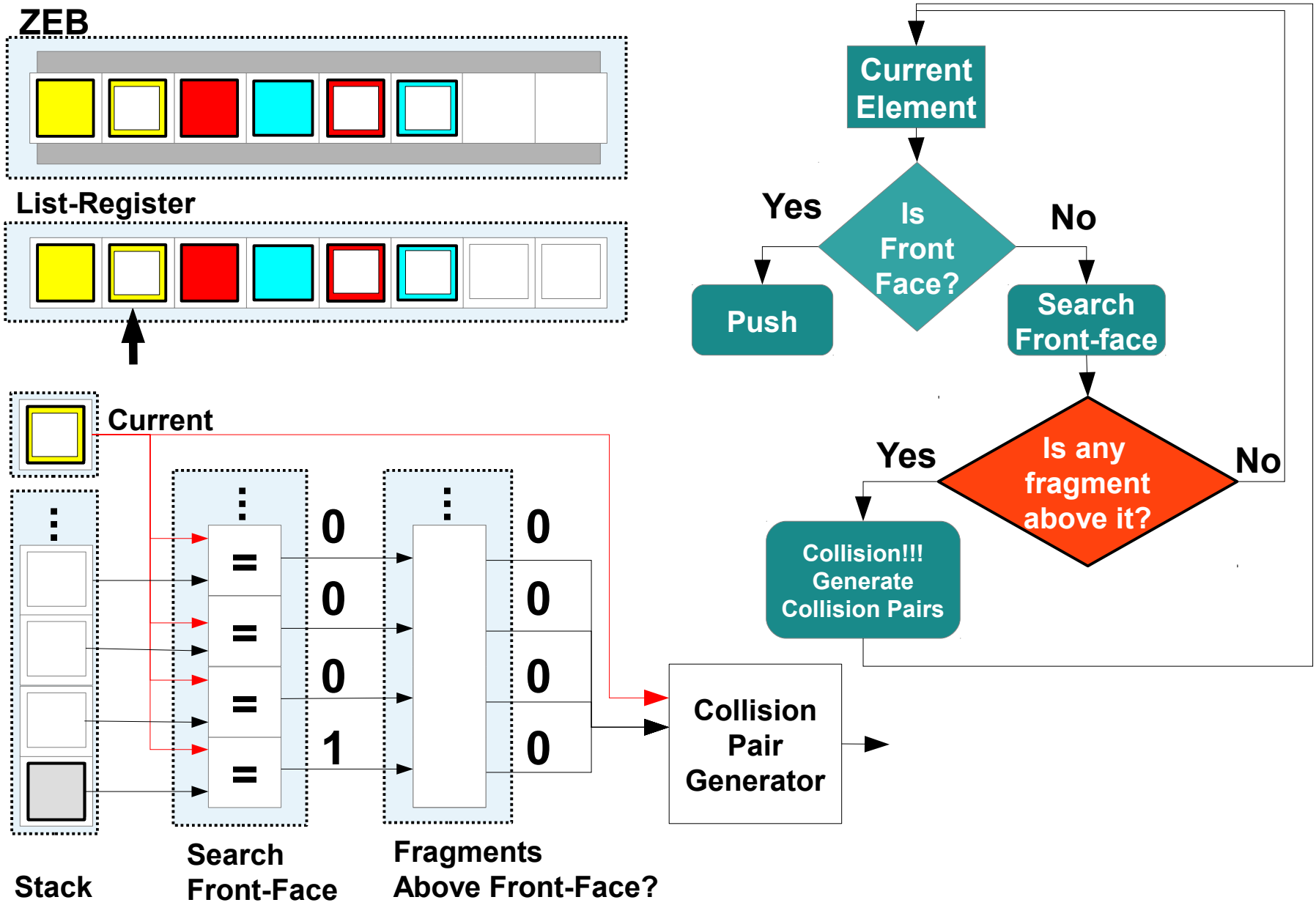
Z-Overlap Test



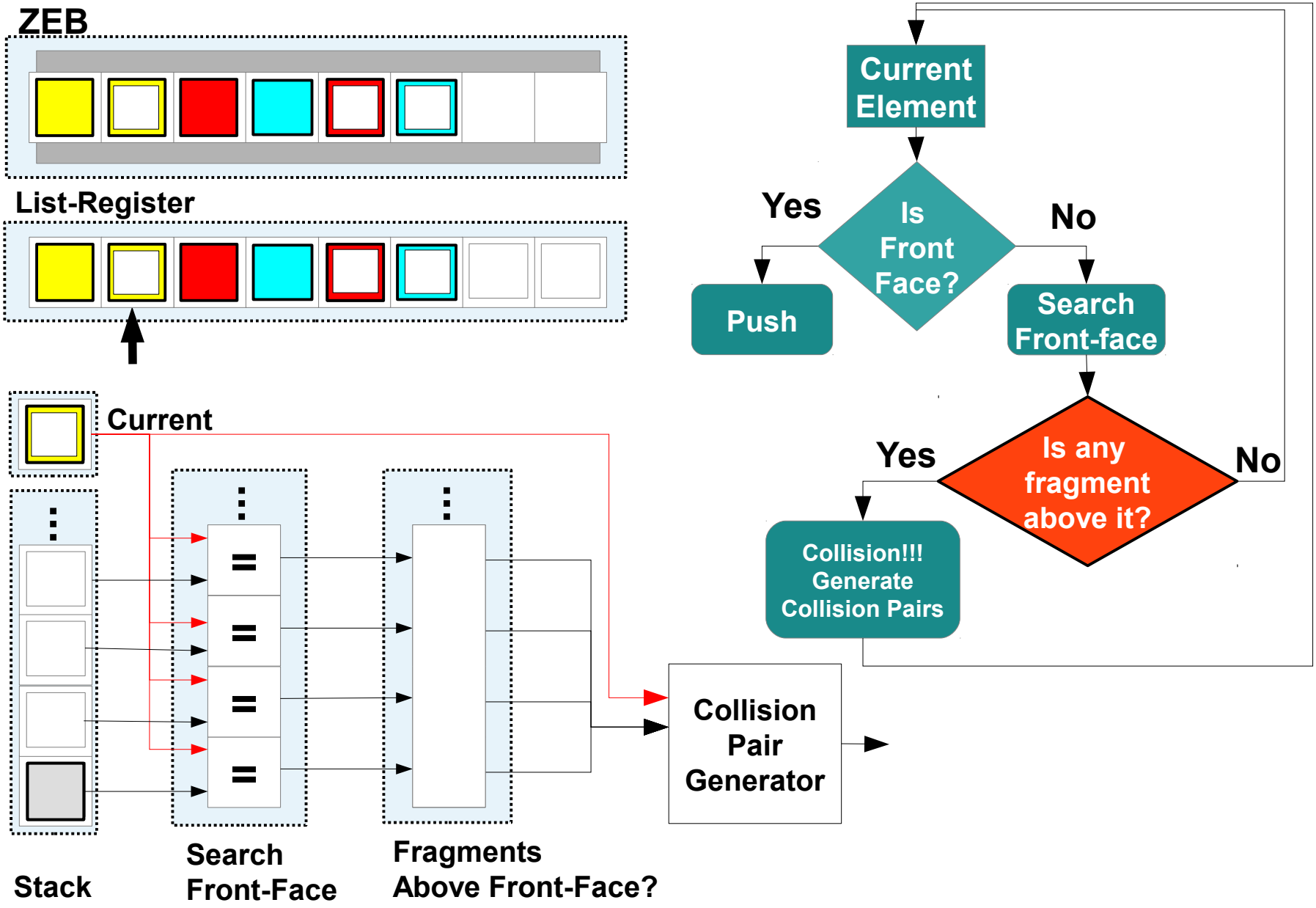
Z-Overlap Test



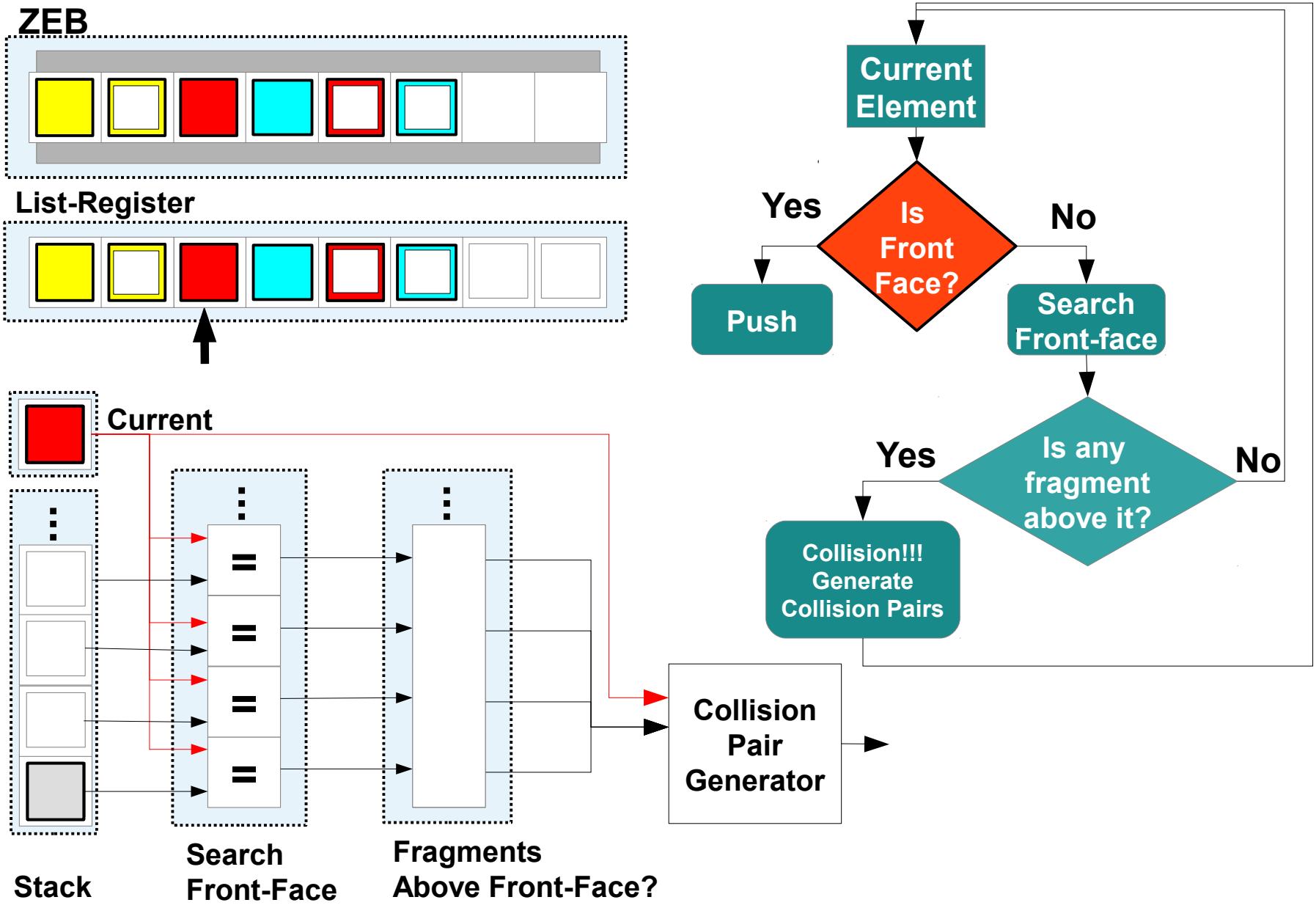
Z-Overlap Test



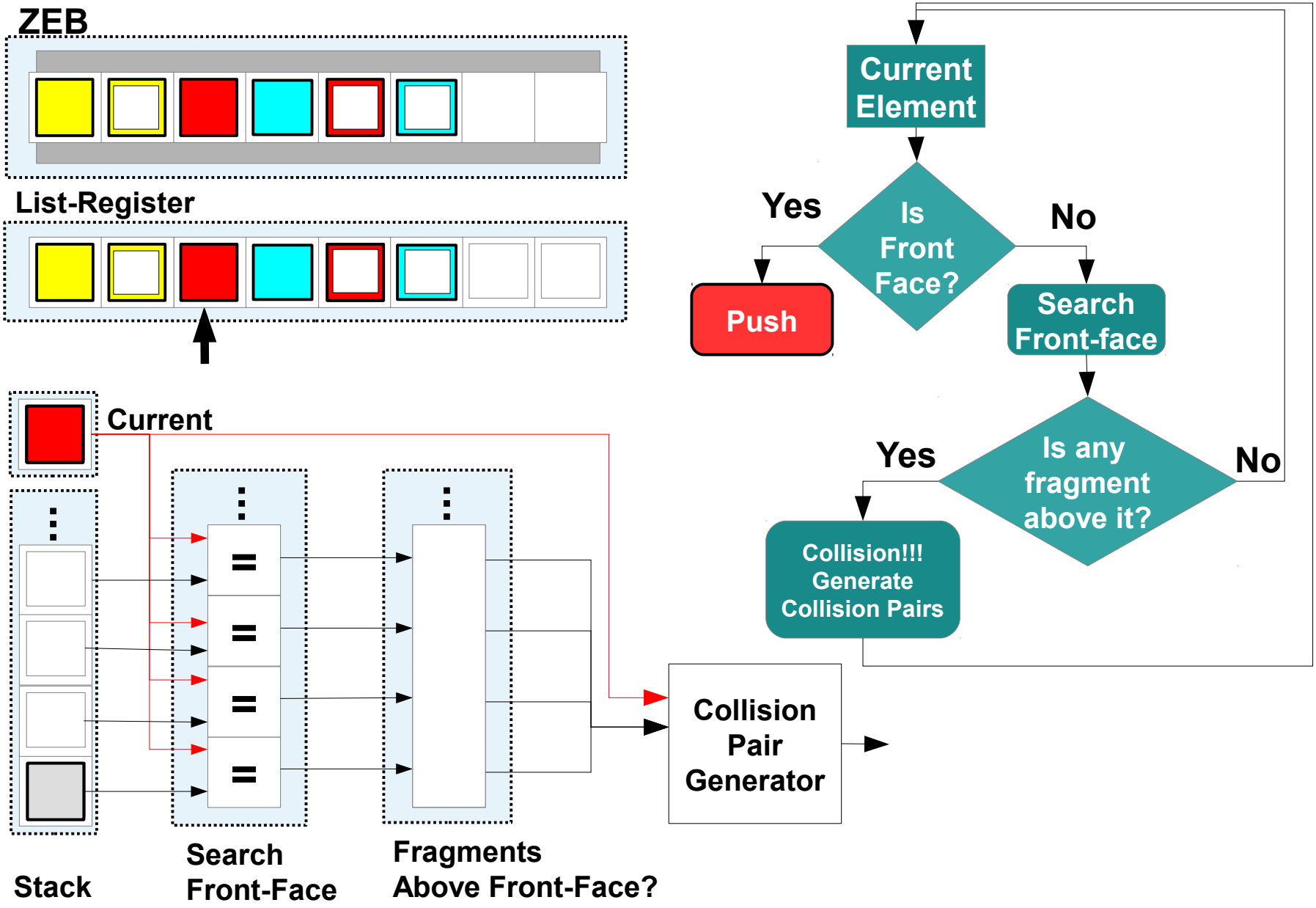
Z-Overlap Test



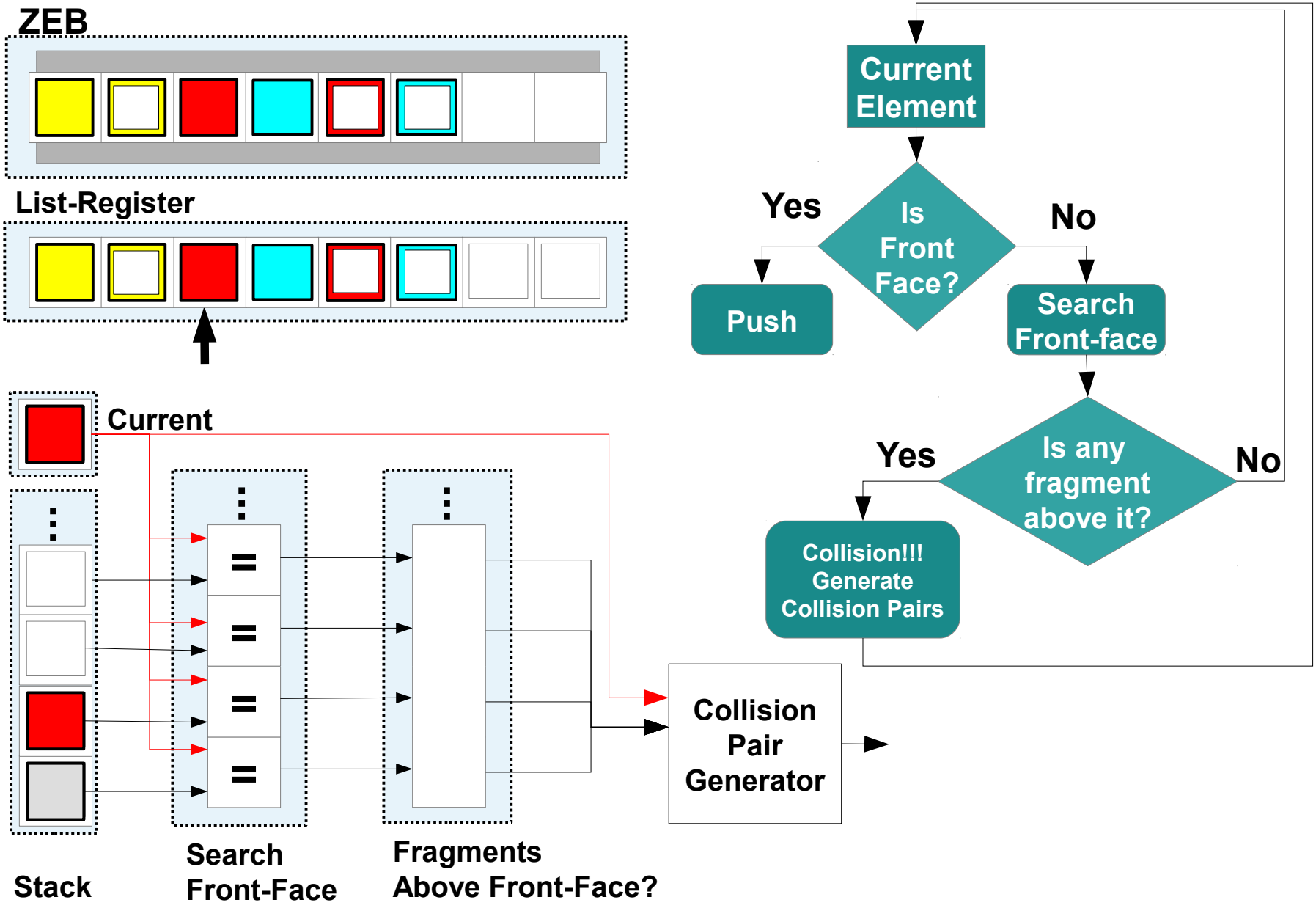
Z-Overlap Test



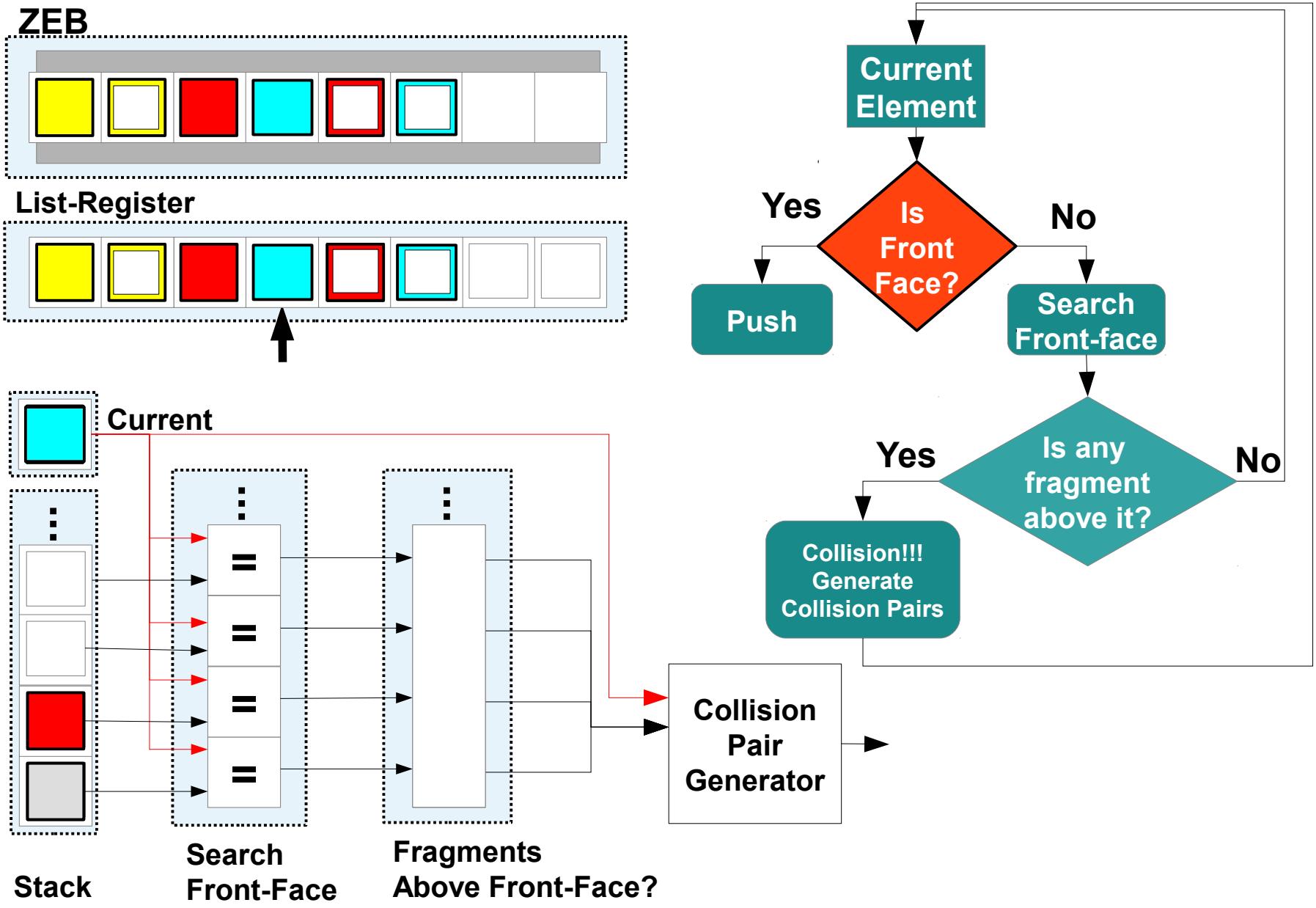
Z-Overlap Test



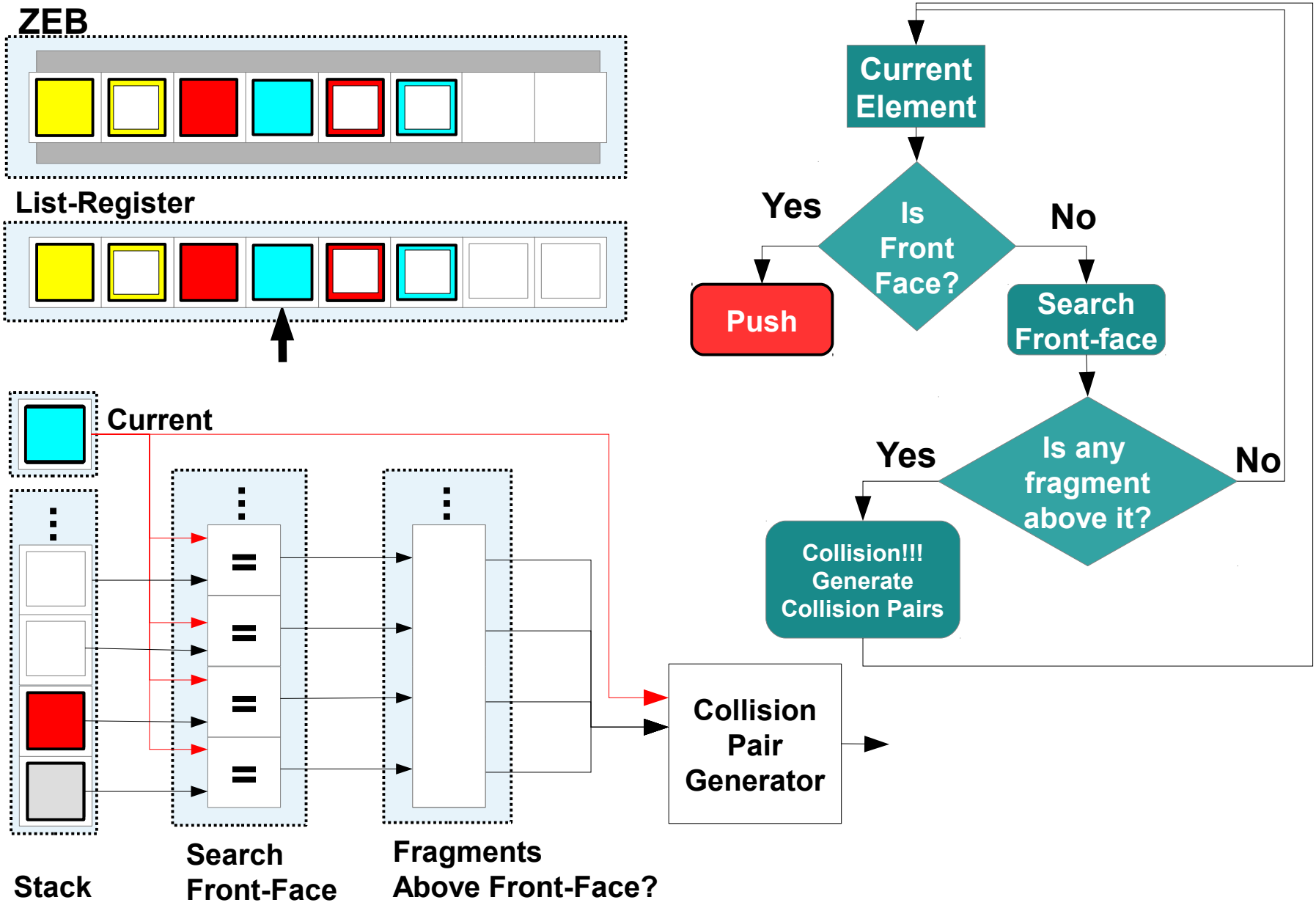
Z-Overlap Test



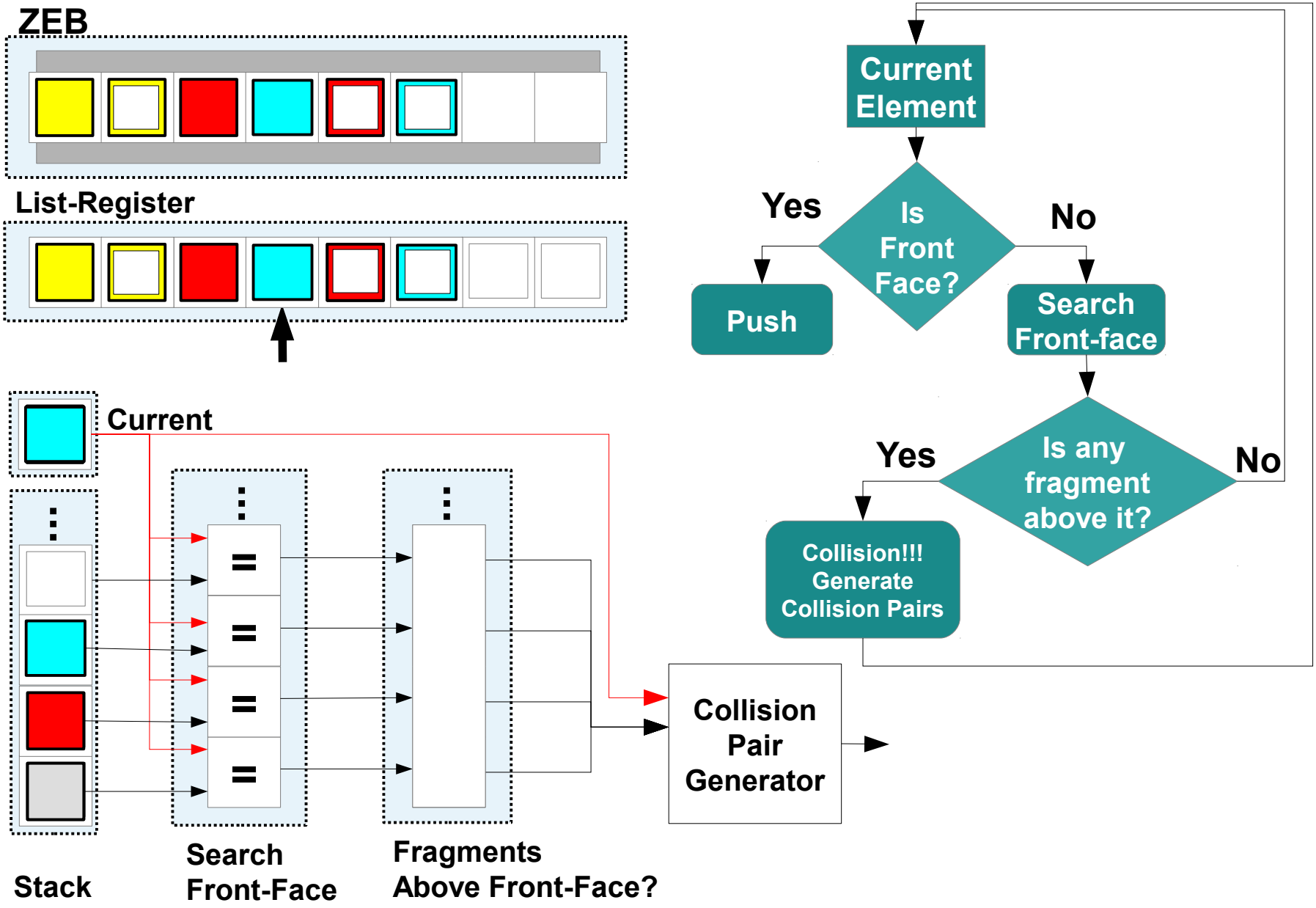
Z-Overlap Test



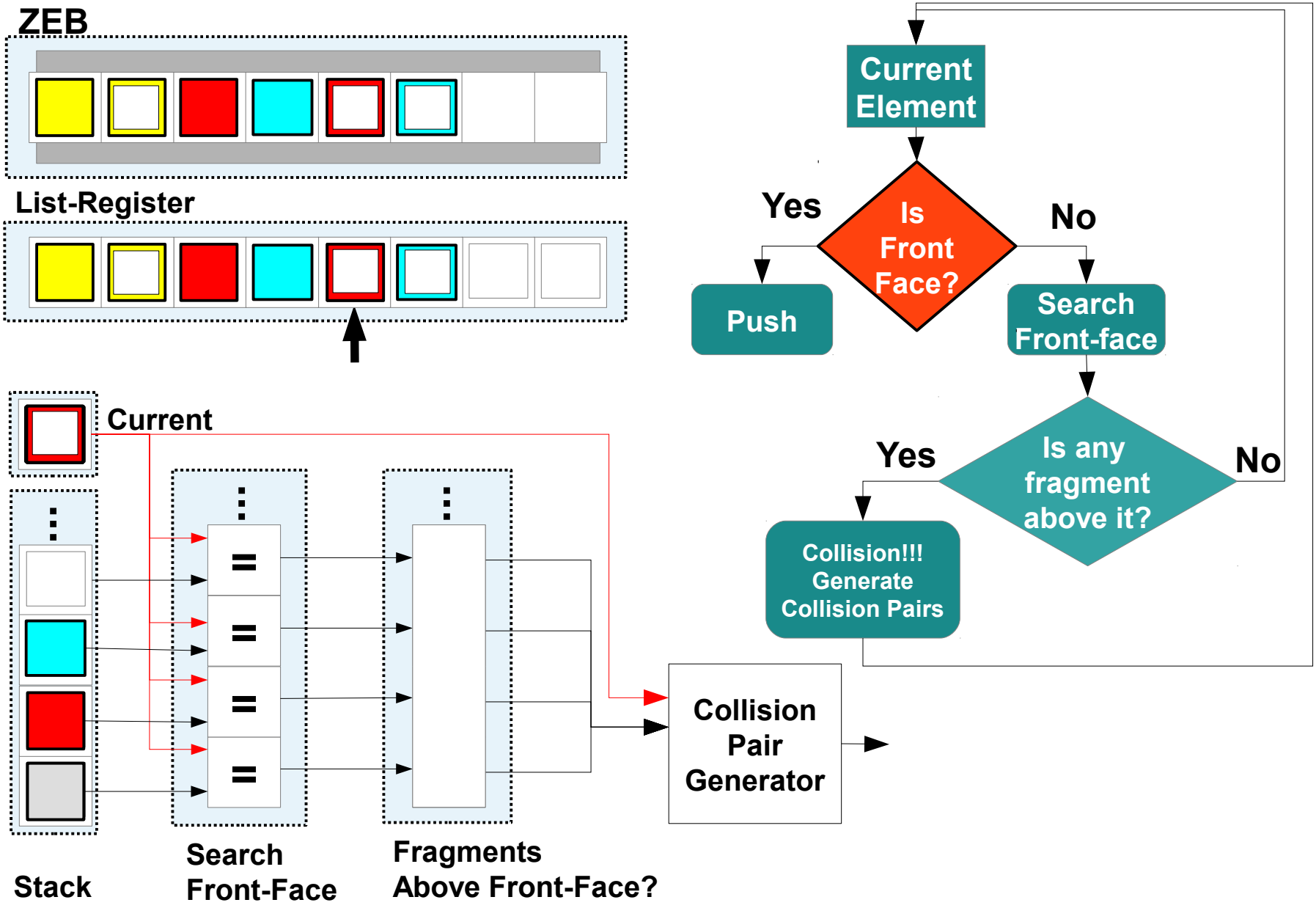
Z-Overlap Test



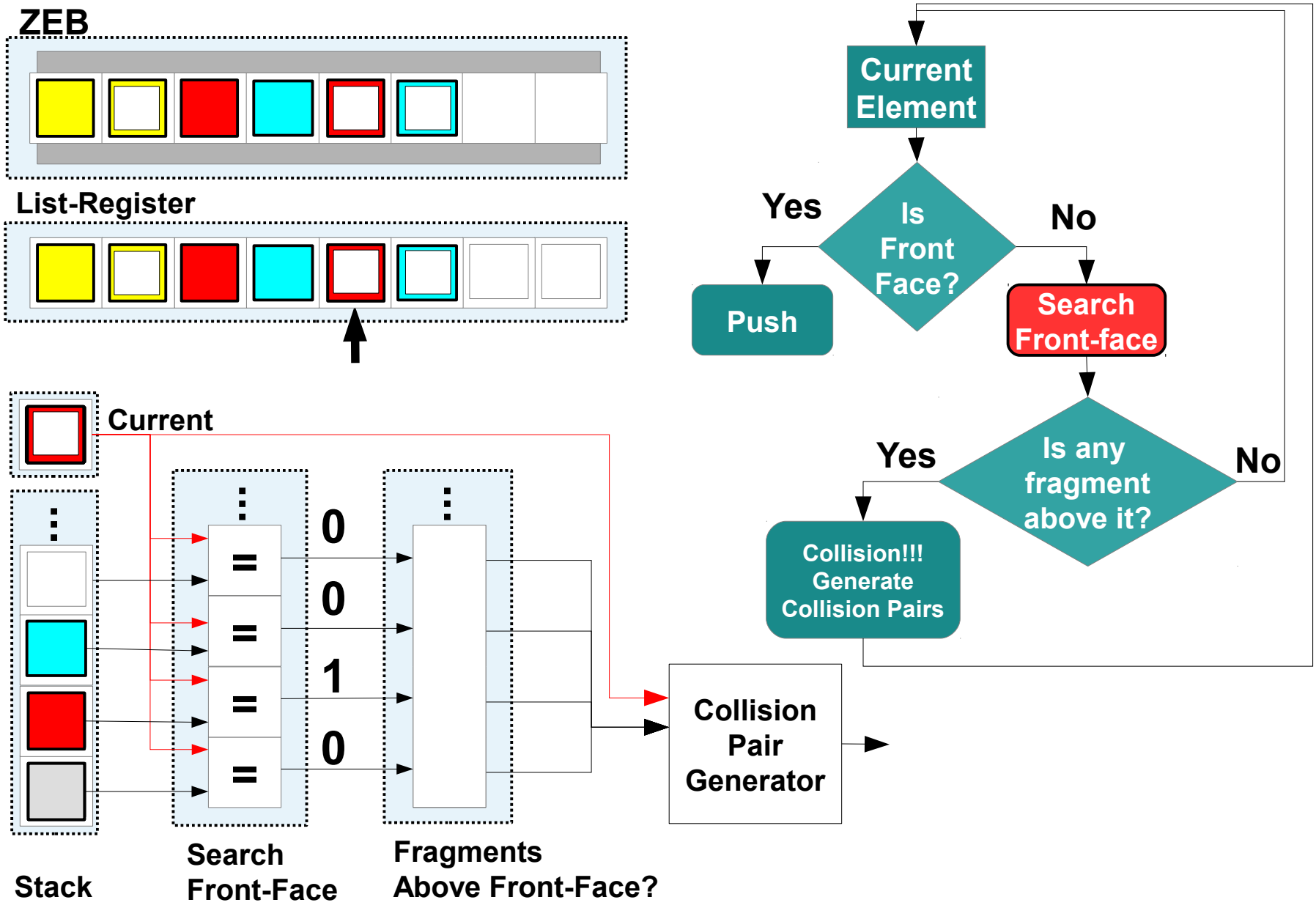
Z-Overlap Test



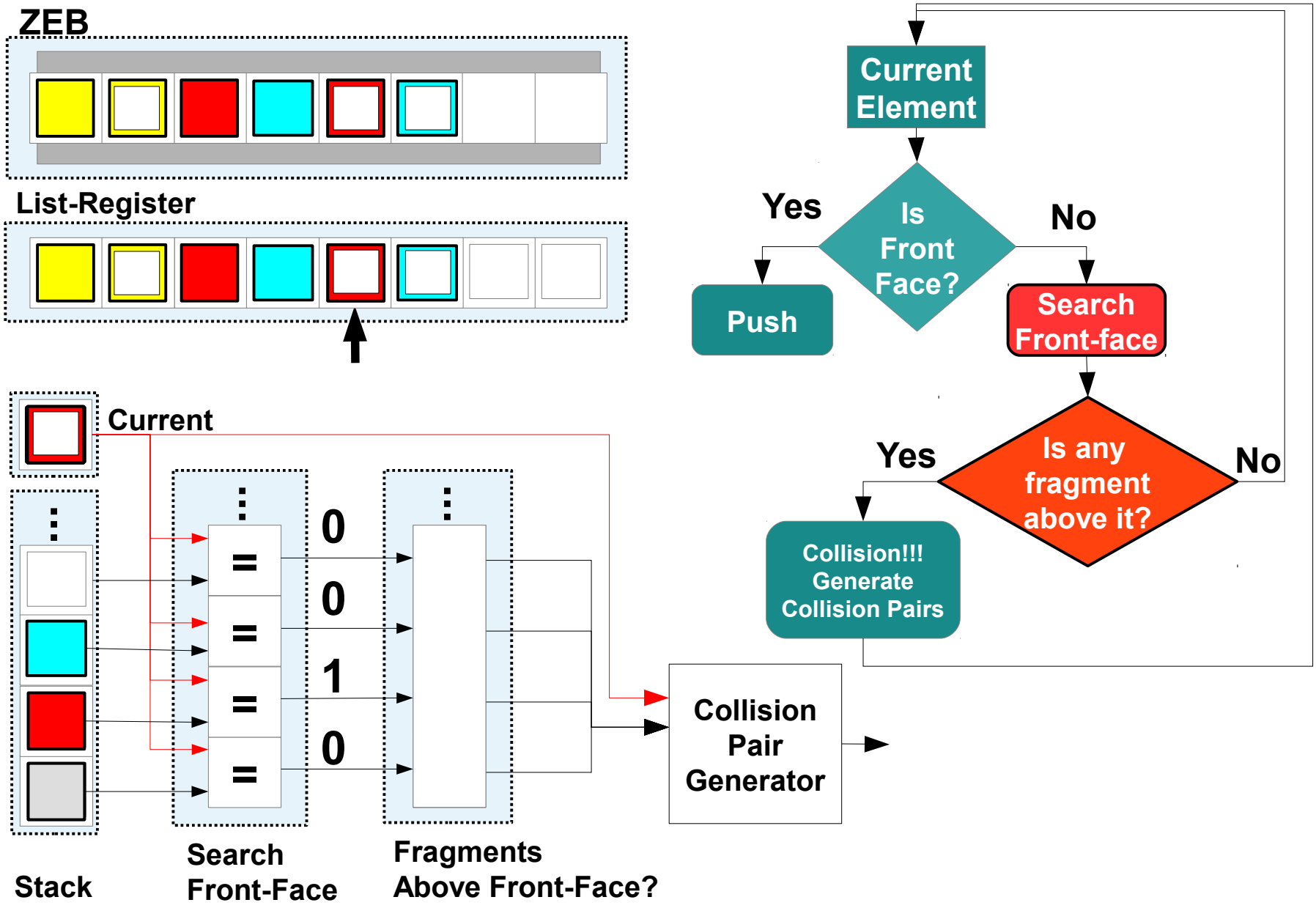
Z-Overlap Test



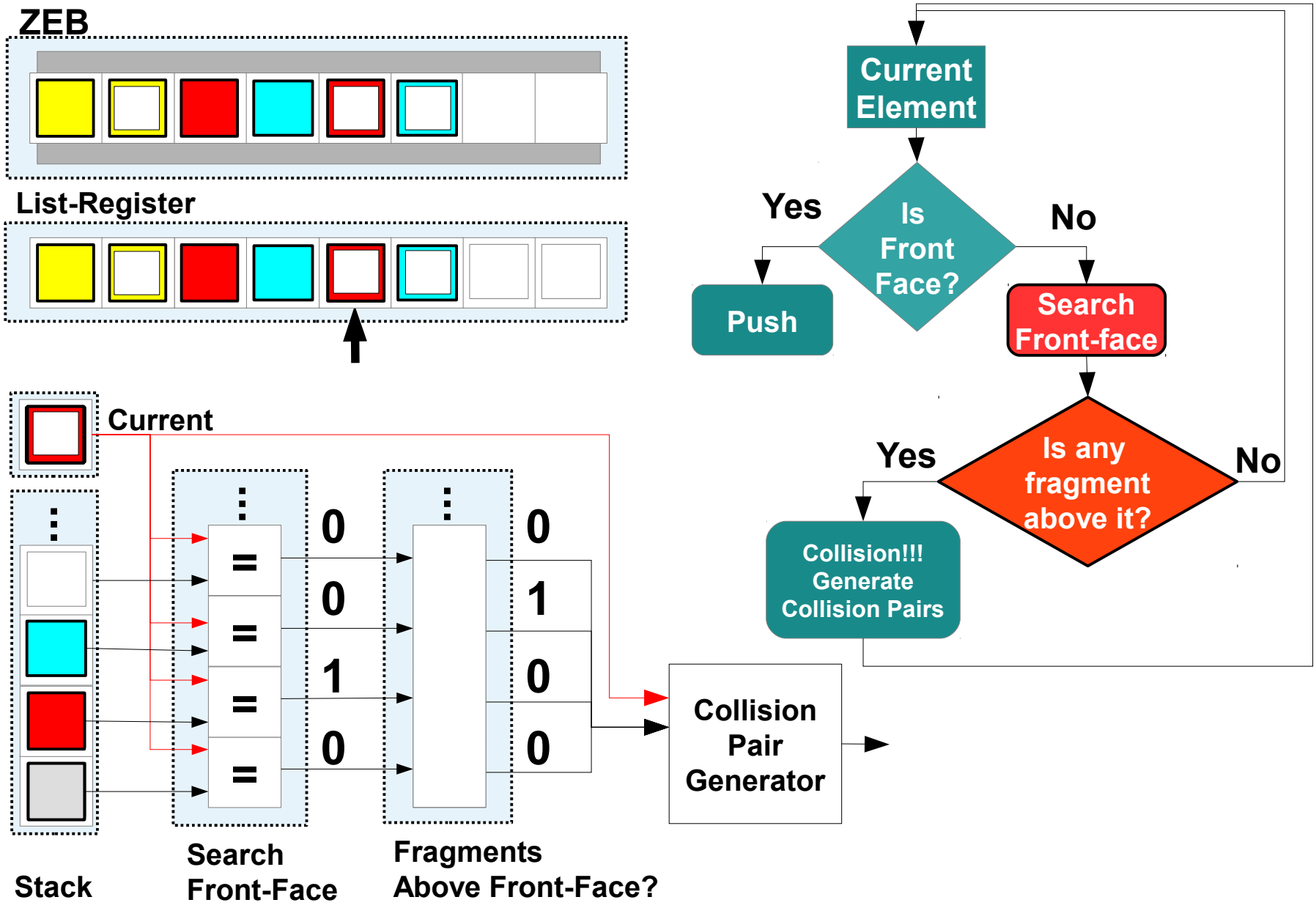
Z-Overlap Test



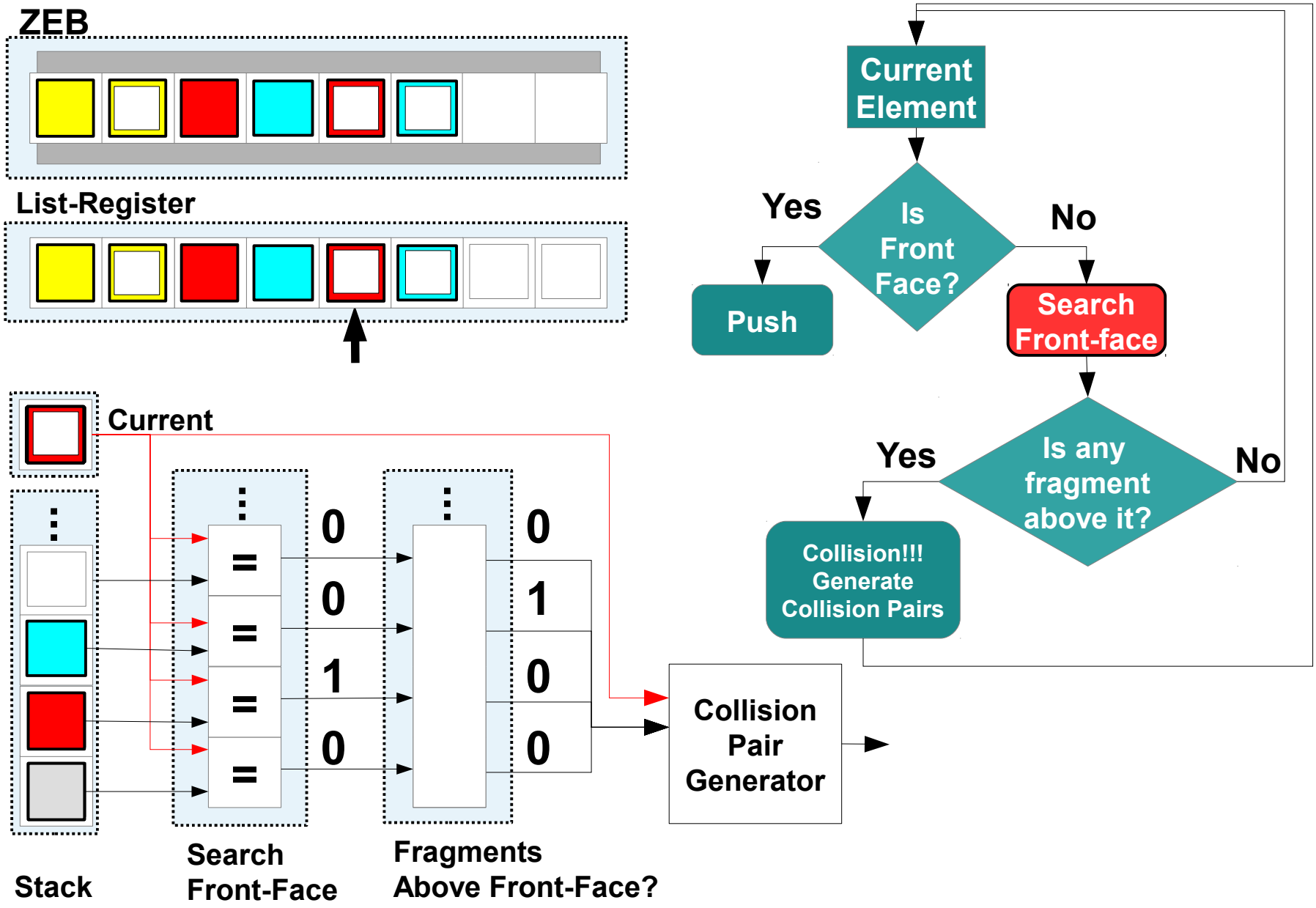
Z-Overlap Test



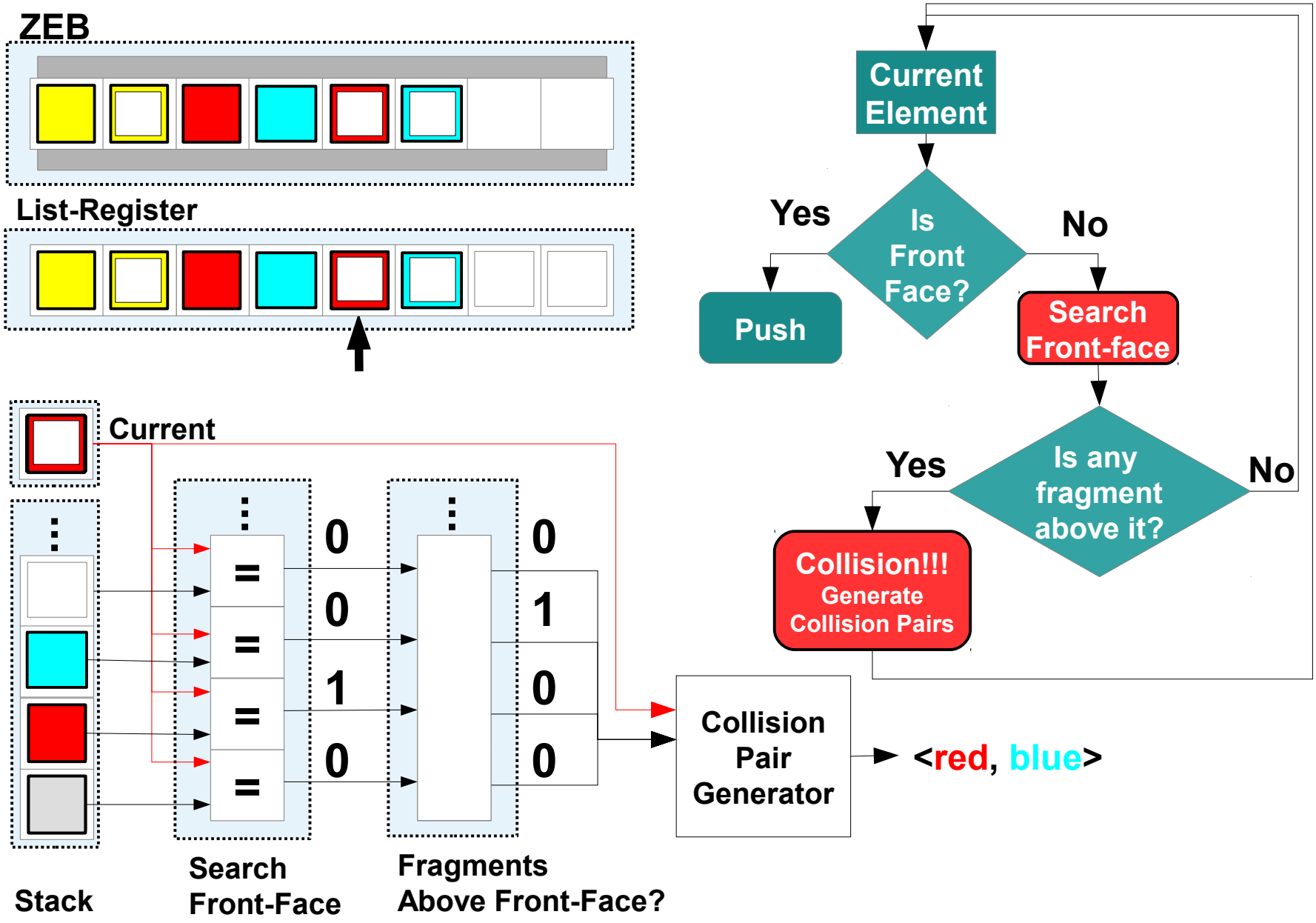
Z-Overlap Test



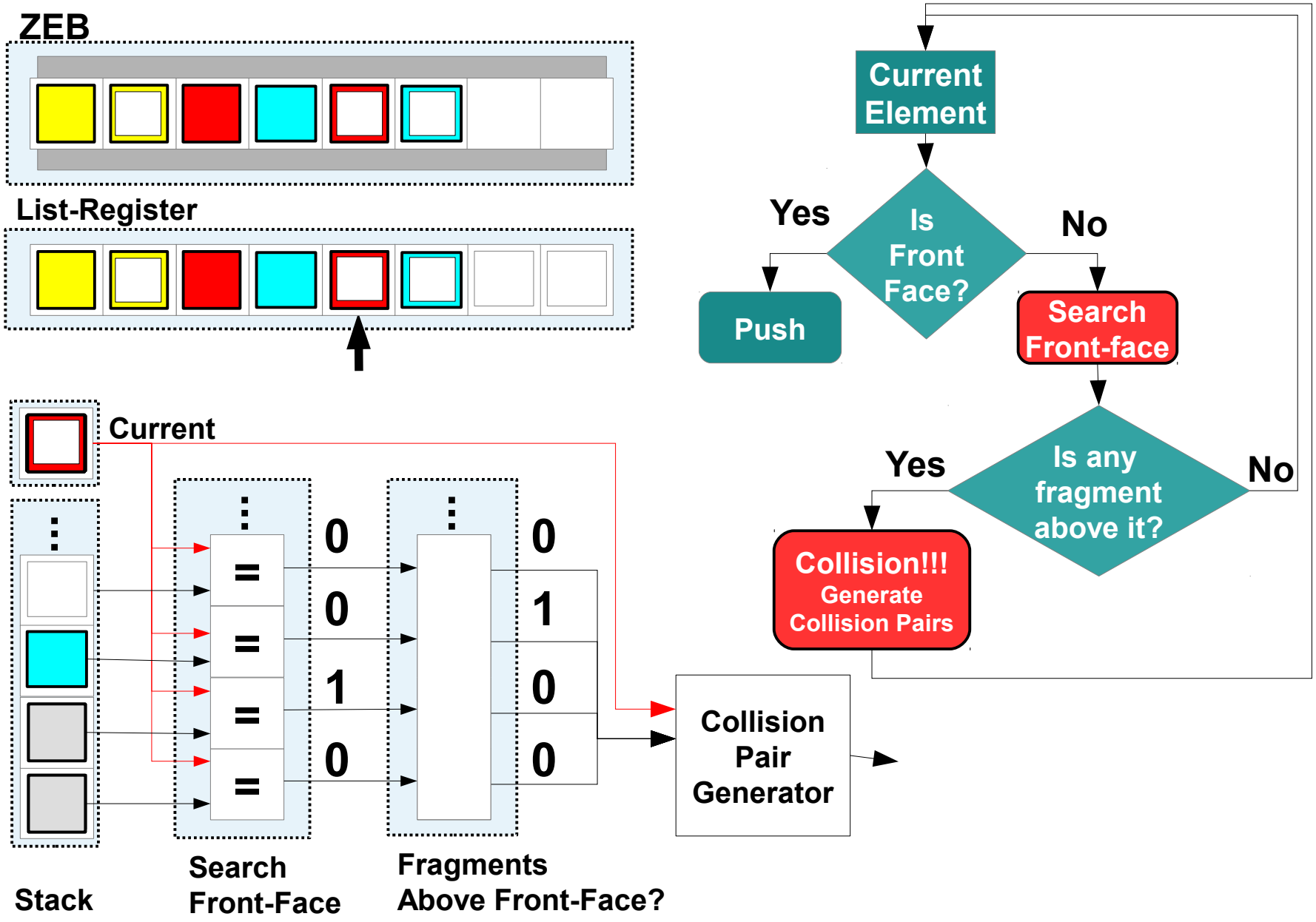
Z-Overlap Test



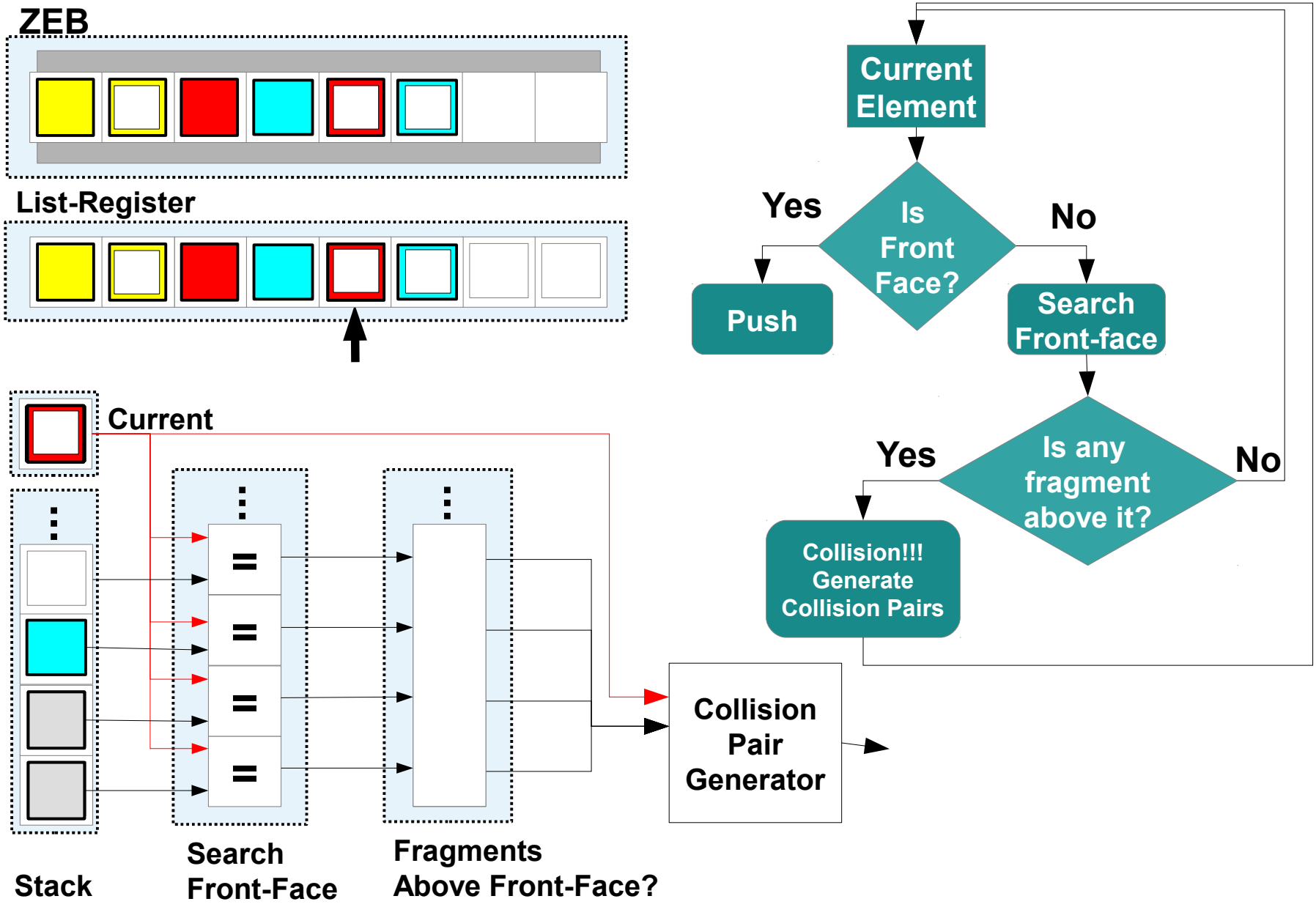
Z-Overlap Test



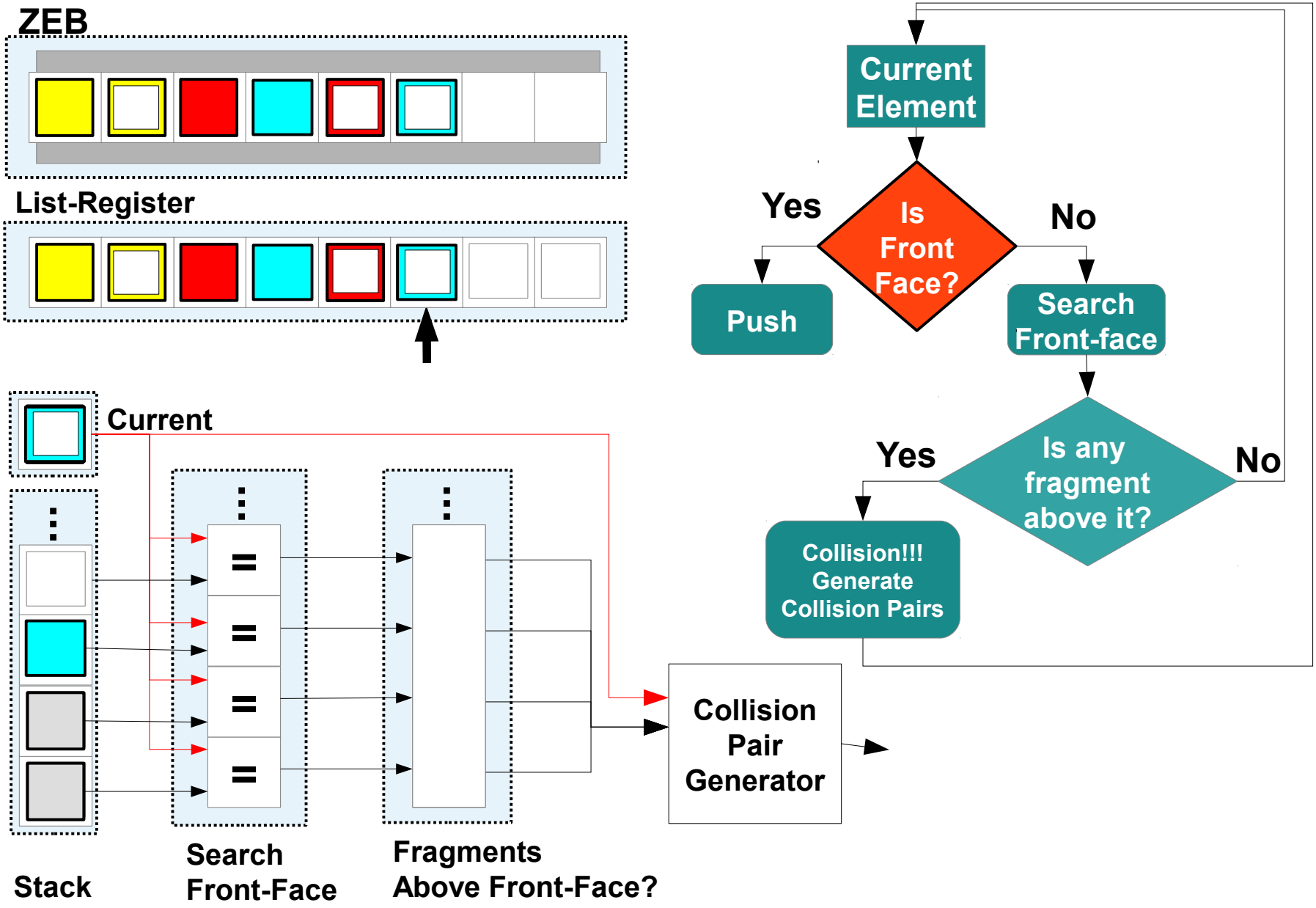
Z-Overlap Test



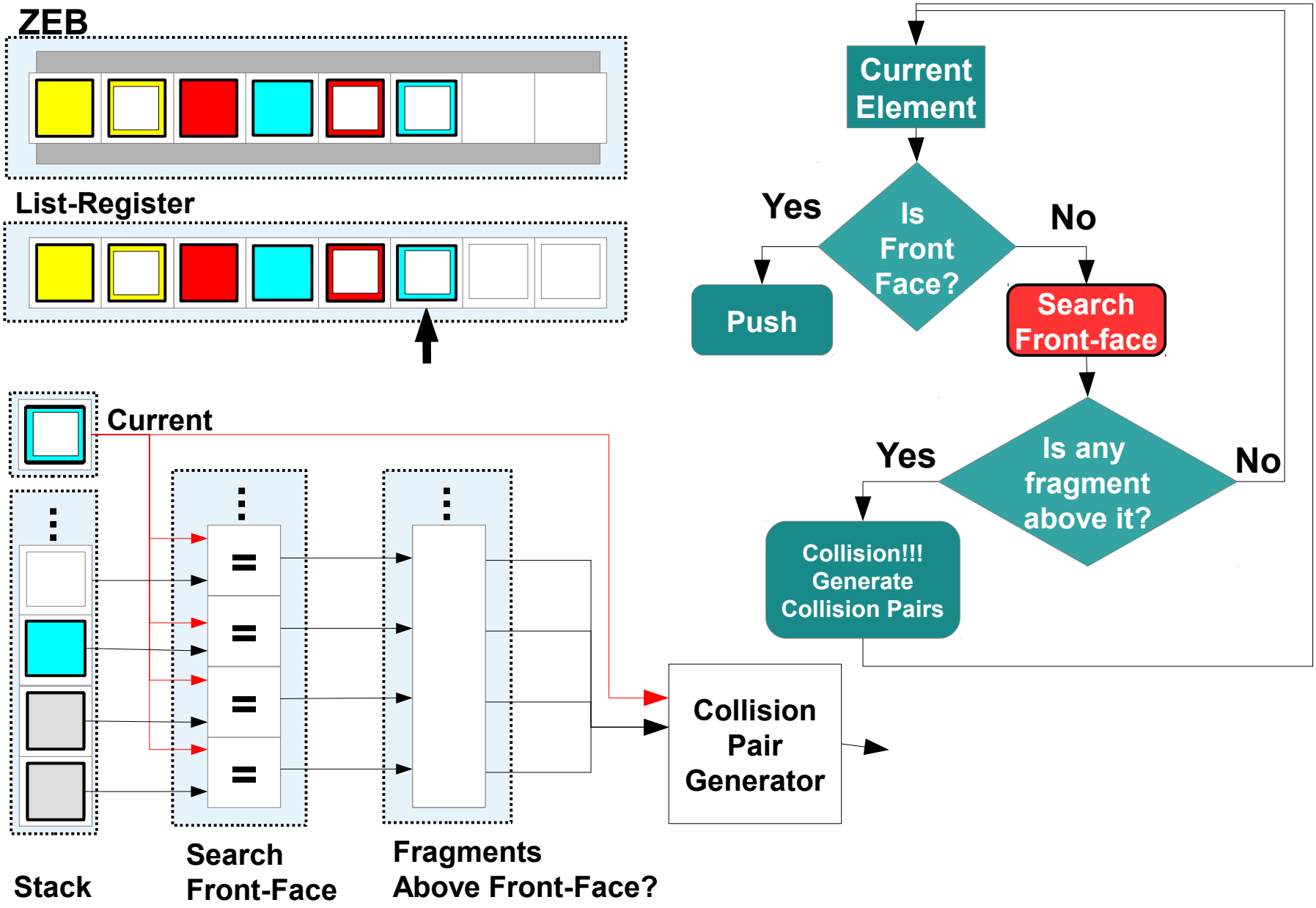
Z-Overlap Test



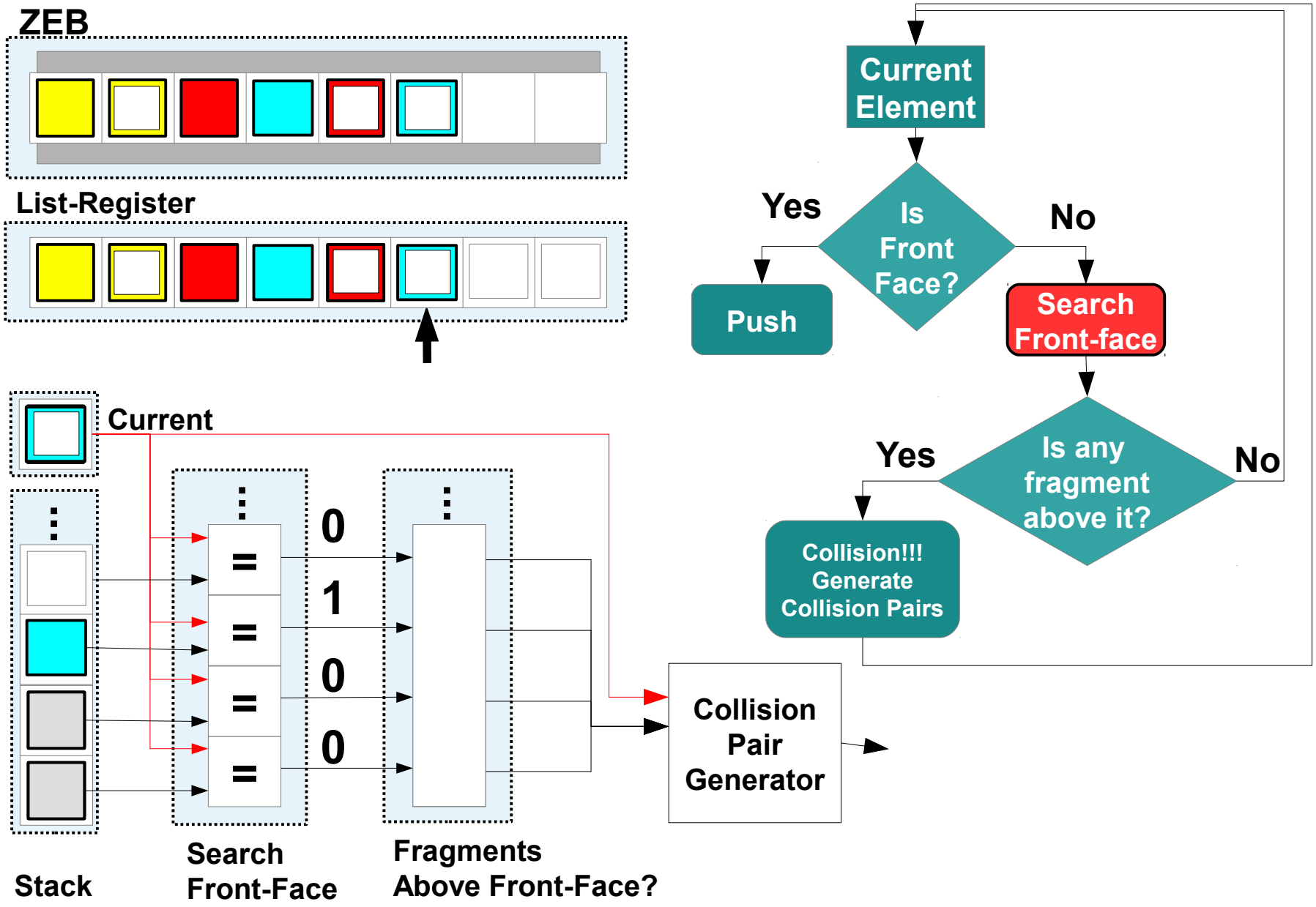
Z-Overlap Test



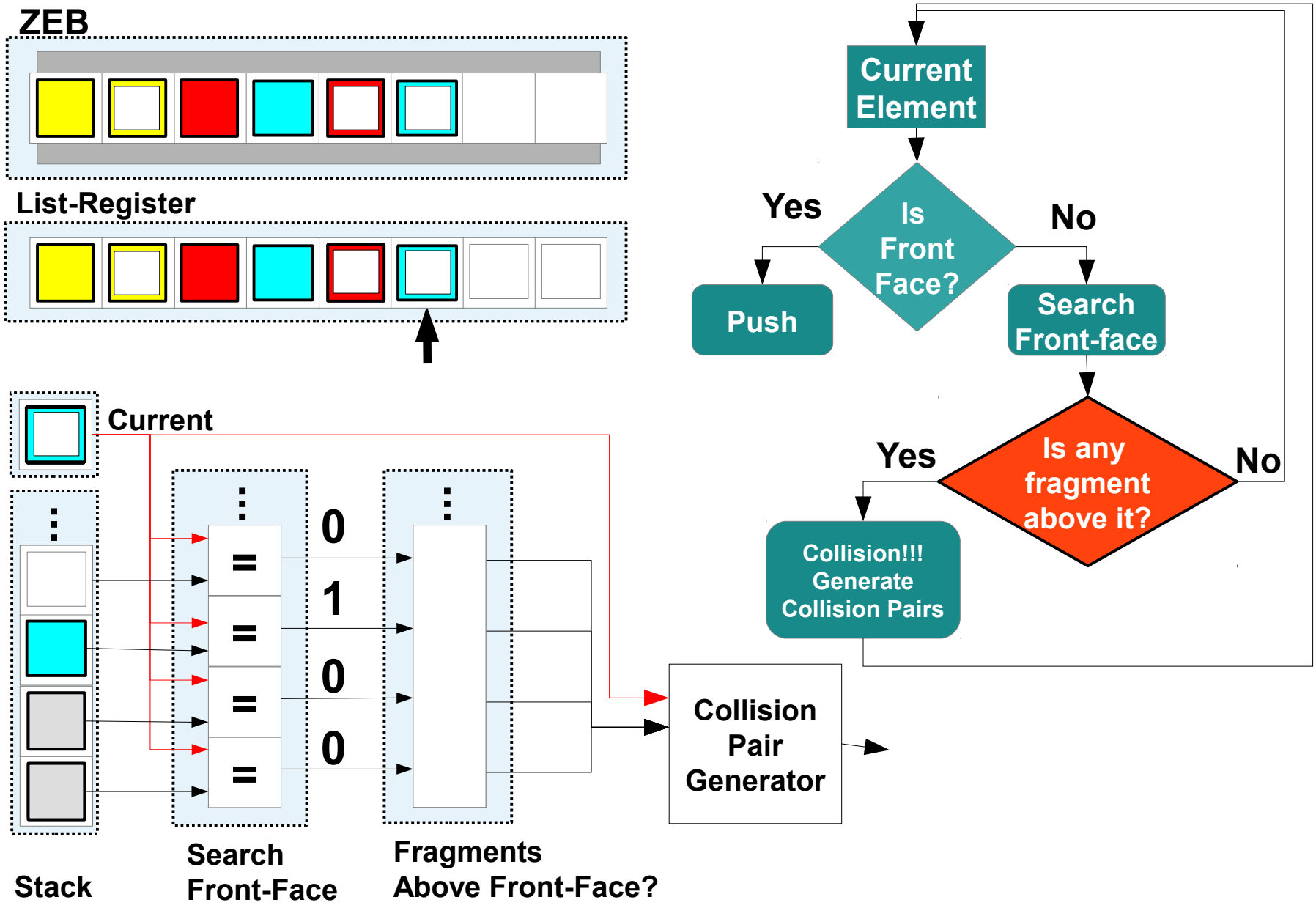
Z-Overlap Test



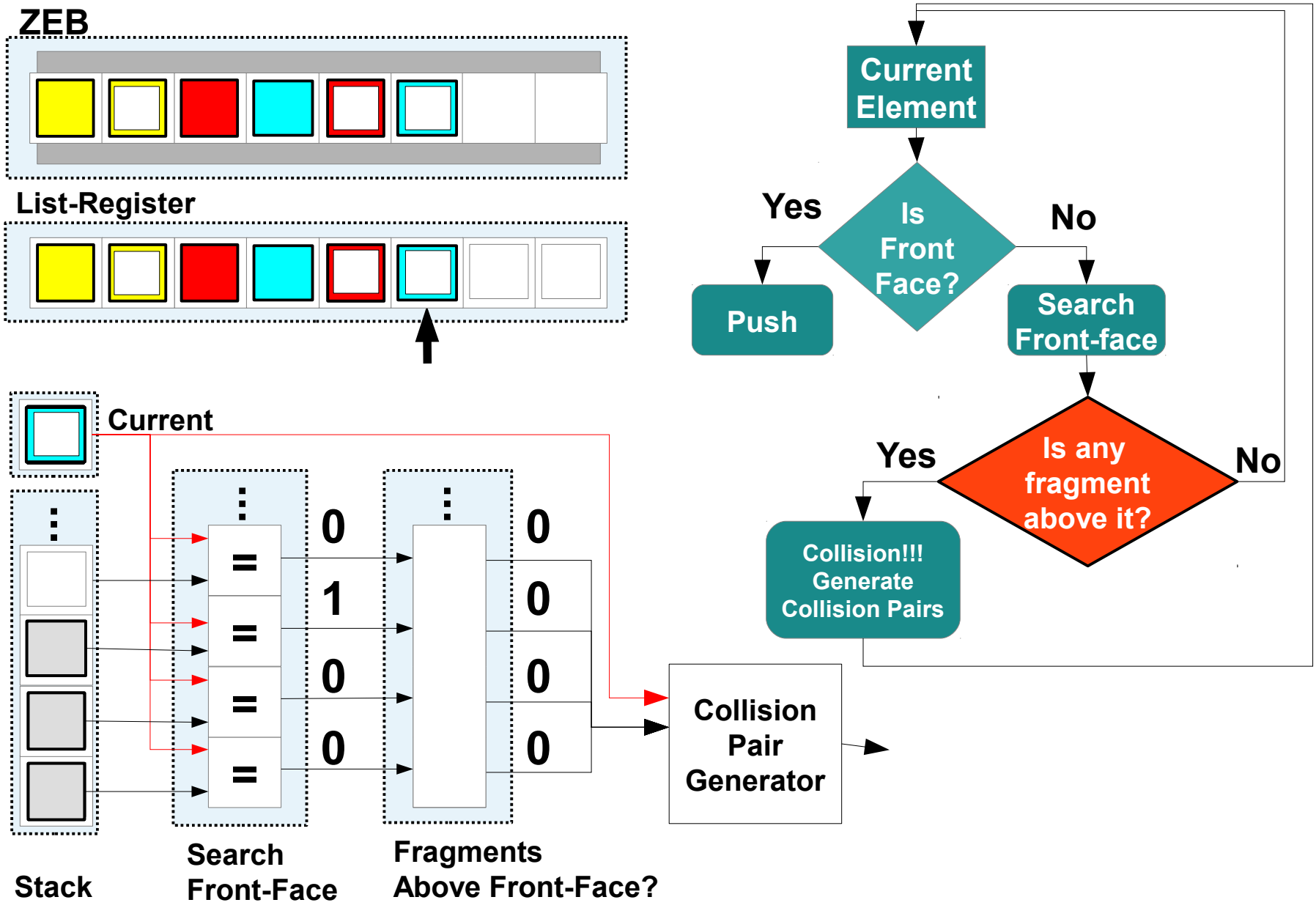
Z-Overlap Test



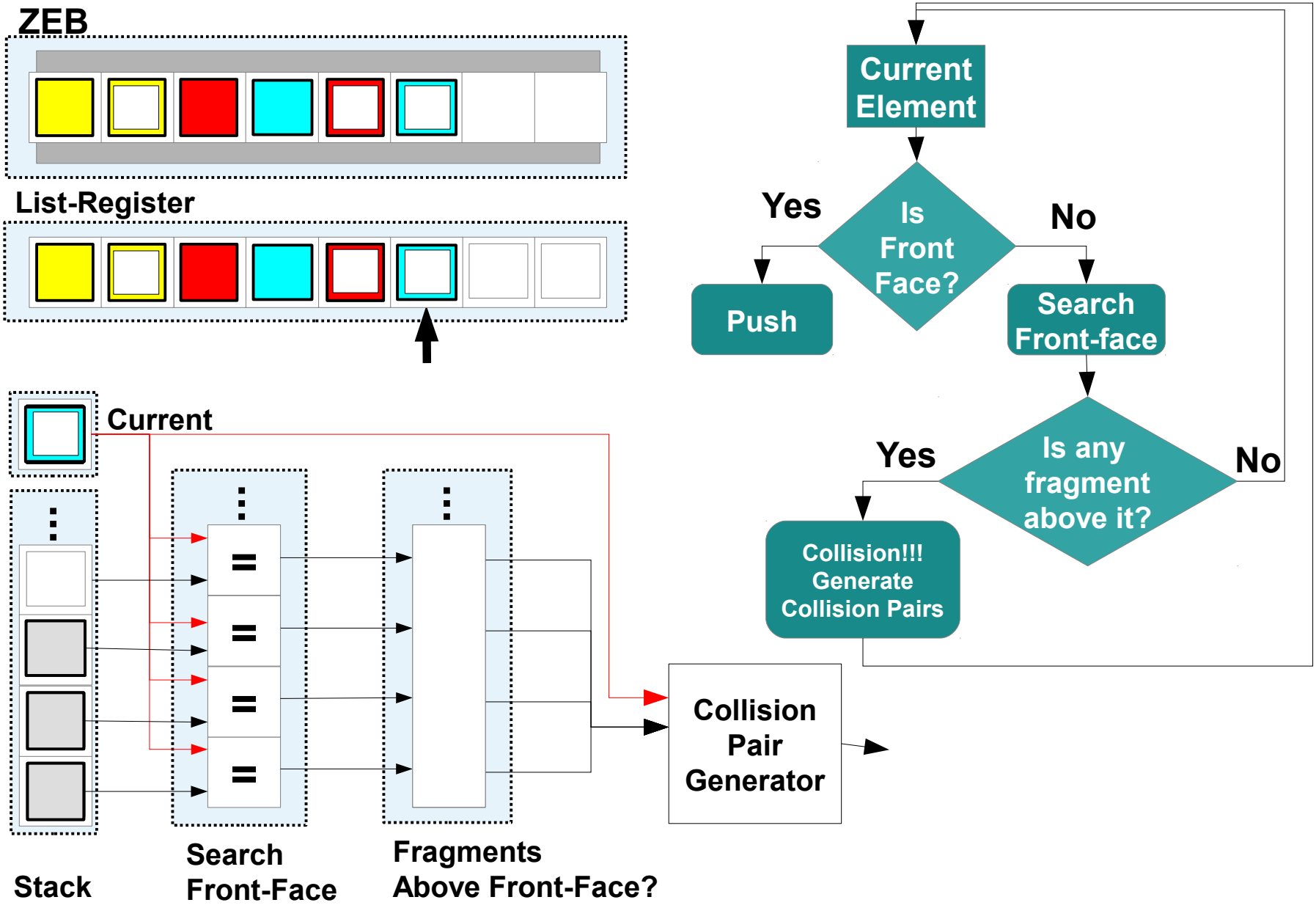
Z-Overlap Test



Z-Overlap Test



Z-Overlap Test



Outline

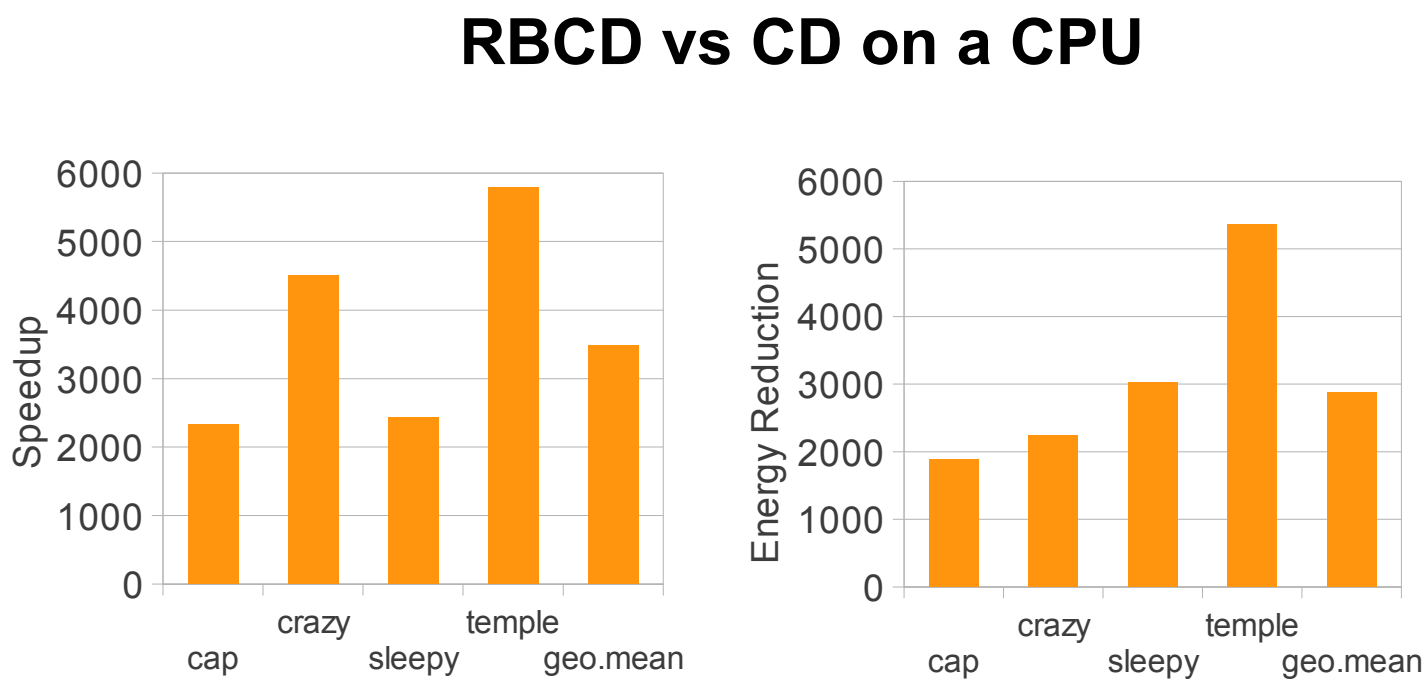
1. Motivation
2. Collision Detection (CD)
3. Render-Based CD in the GPU
4. **Results**
5. Conclusion

Evaluation Methodology

- **TEAPOT** simulation infrastructure
  - **Android** and **OpenGL ES**
  - GPU timing simulator models:
    - Tile-Based Rendering architecture (**ARM Mali 400MP-like**)
  - GPU power model based on **McPAT**
    - Extended with **RBCD unit**
- **Marss** cycle accurate full system simulator
- Workloads: 4 unmodified Android commercial games

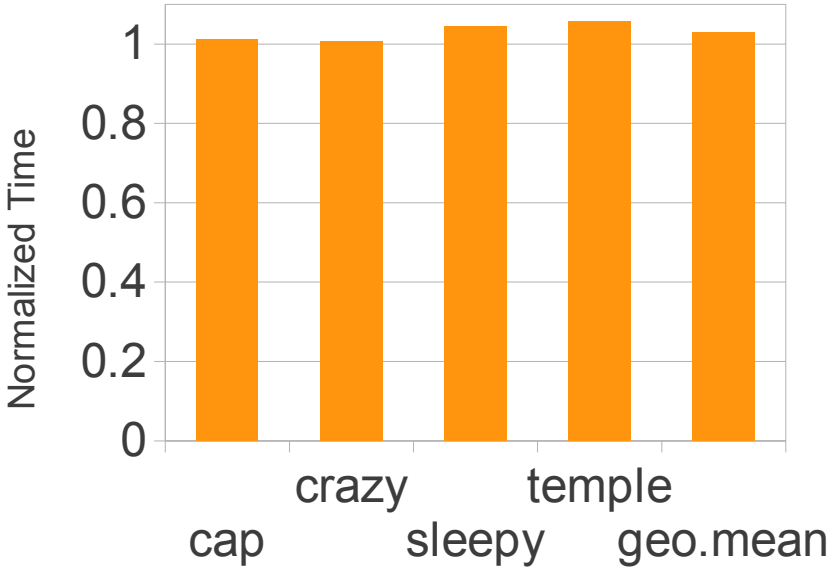
Benchmark	Type	Donwnloads
Captain America	beat'em up	10-50 M
Crazy Snowboard	snowboard arcade	5-10 M
Sleepy Jack	action	100-500 K
Temple Run	adventure arcade	100-500 M

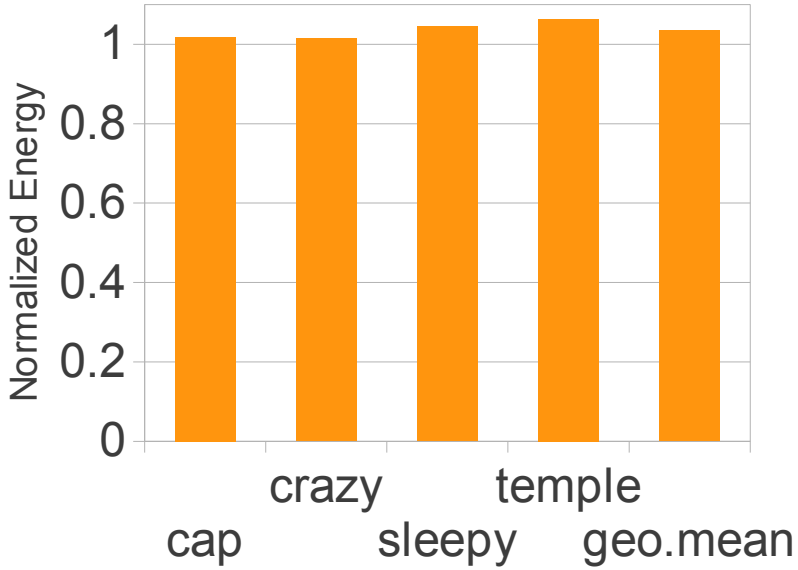
Performance and Energy of CD



- **3400x** speedup and **2875x** energy reduction on average
  - RBCD reuses rendering results

GPU Overhead

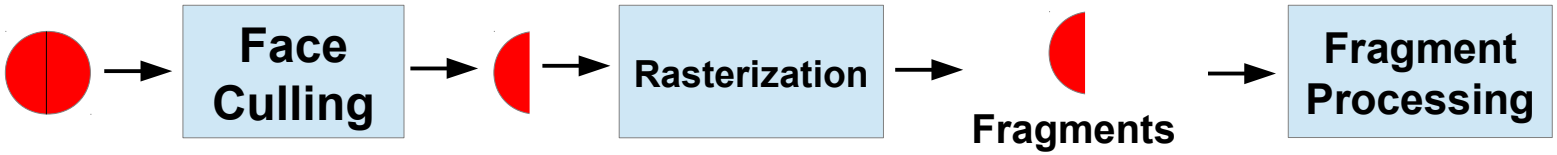




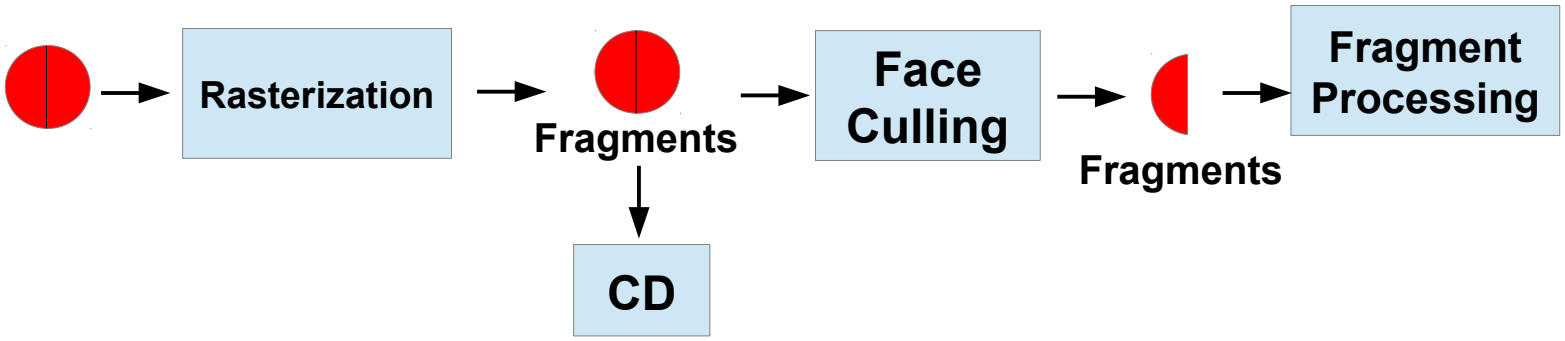
- Time overhead: **3%**
- Energy: **3.5%**
- Area overhead: < **1%**

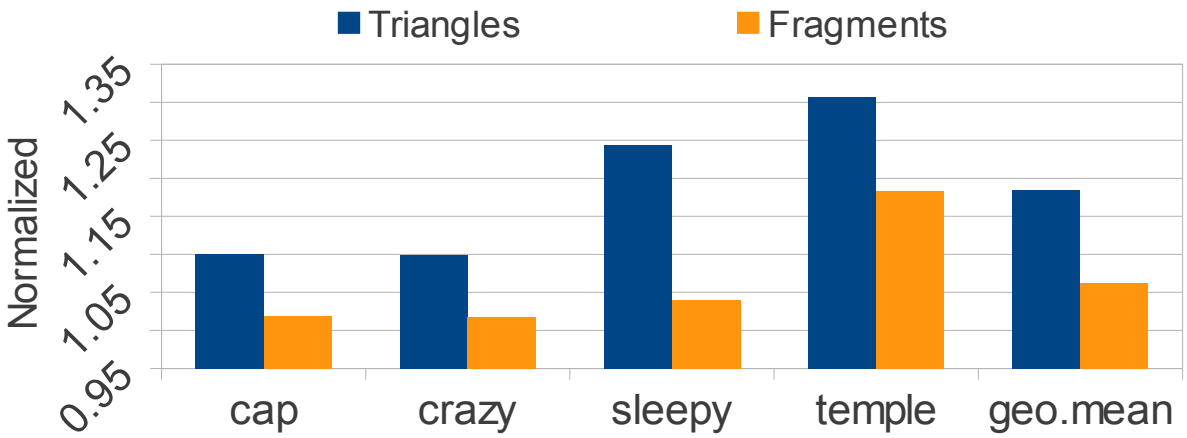
GPU Overhead: Face Culling

Common Face Culling:



Deferred Face Culling:





Rasterize 18% more triangles, 6% more fragments

Conclusions

- Energy budget limits the **quality** and **realism** of CD in mobile graphics animation applications
- Most of the computation required by Image-Based CD is **already done** in image rendering
- RBCD provides
  - **2875x** energy reduction
  - **3400x** speedup
  - Pixel level accuracy
  - Small overheads: time (**3%**), energy (**3.5%**) and area (**1%**)
- RBCD is a **low-energy** yet **high-fidelity** CD solution

48th International Symposium on Microarchitecture

Ultra-Low Power Render-Based Collision Detection for CPU/GPU Systems

Enrique de Lucas
Joan-Manuel Parcerisa
Pedro Marcuello
Antonio González

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH