블랙홀 실험실

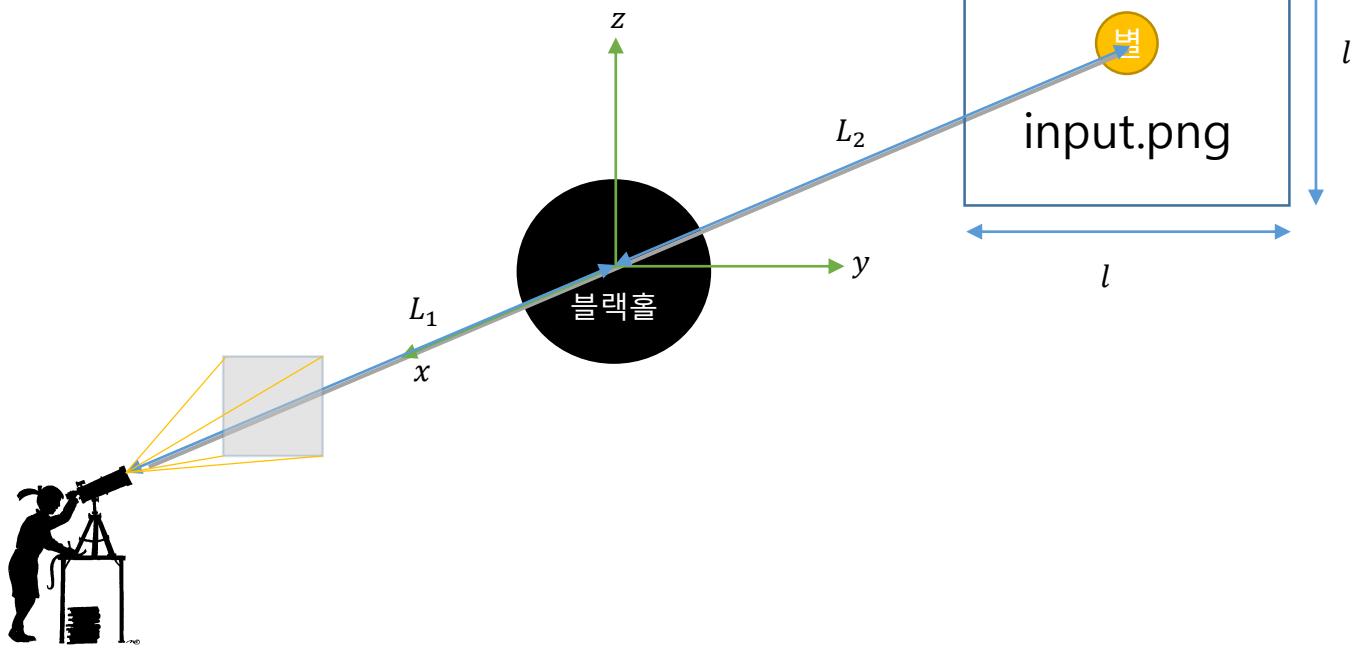
국가슈퍼컴퓨팅 청소년캠프

병렬 프로그래밍 과제

2016.08.02.

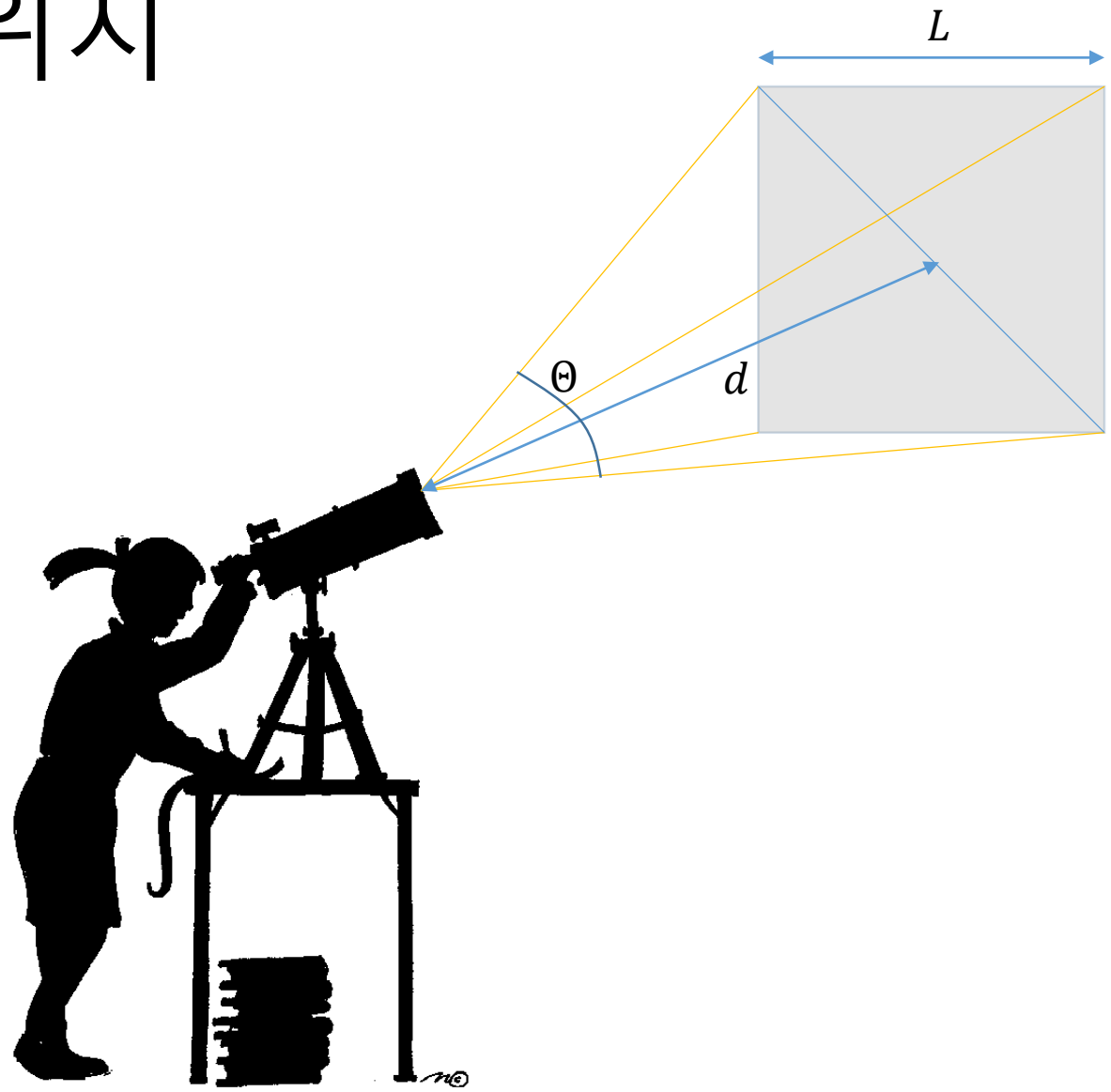
실험실 구조

- $L_1 = 10M$: 블랙홀과 관측자 사이의 거리
- $L_2 = 10M$: 블랙홀과 그림 사이의 거리
- $l = 10M$: 그림의 크기



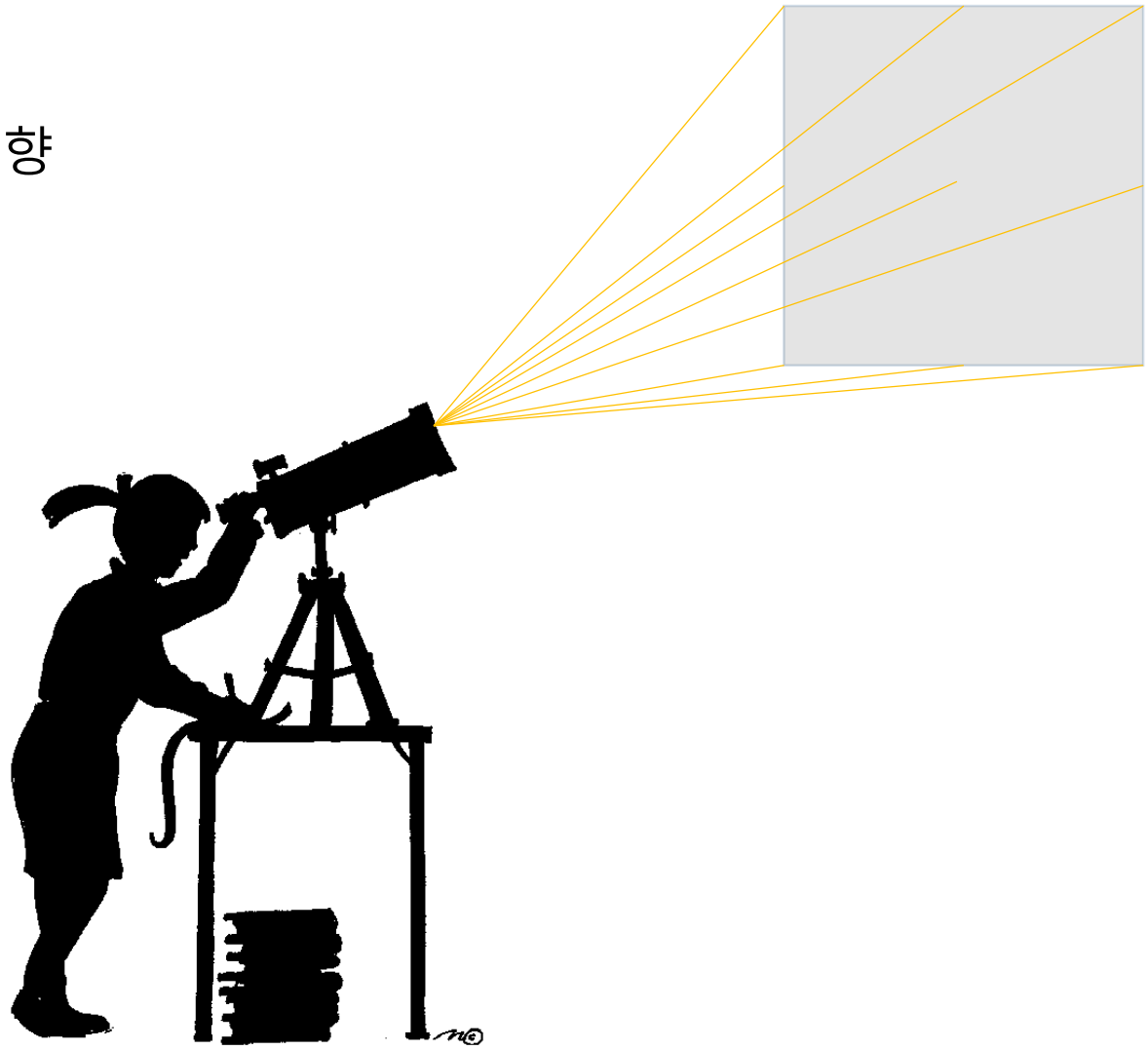
관찰자가 보는 상의 위치

- L : 상의 크기
- d : 상의 위치
- $\Theta = 120°$: 시야각
- $\tan\left(\frac{\Theta}{2}\right) = \frac{L}{\sqrt{2}d}$



시뮬레이션 방법

- 상의 각각의 픽셀 위치를 통과하는 다양한 방향의 빛 다발을 고려
- 각각의 빛 다발이 어느 위치에서 온 것인지를 시뮬레이션으로 계산
- 그림으로부터 온 경우 그림의 이미지를 표시
- 블랙홀로부터 온 경우 검은색으로 표시
- 그 외의 경우 흰색으로 표시



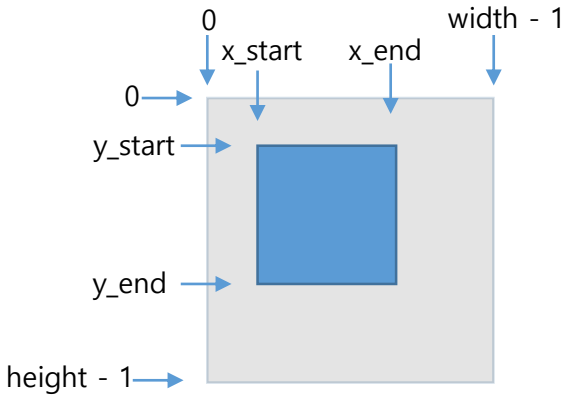
코드 설명 (main.c) : 상수 정의

- // Information for view
- **#define C_VISUAL_ANGLE 120.**
- // Information for time evolution
- **#define C_TIME_STEP 20000**
- **#define C_TIME_INTERVAL 0.05**
- // Information for images
- **#define C_IMPORT_IMAGE_FILENAME "input.png"**
- **#define C_EXPORT_IMAGE_FILENAME "output.png"**
- **#define C_EXPORT_IMAGE_WIDTH 100**
- **#define C_EXPORT_IMAGE_HEIGHT 100 // it should be a multiple of num_procs.**
- **#define C_EXPORT_IMAGE_TOTAL_PIXELS (C_EXPORT_IMAGE_WIDTH * C_EXPORT_IMAGE_HEIGHT)**
- // Information for job division
- **#define C_NUMBER_OF_DIVISION 4**

- C_VISUAL_ANGLE
  - 시야각 Θ
- C_TIME_STEP
  - 총 진행 횟수
- C_TIME_INTERVAL
  - 한번 진행 할 때 마다 흐르는 시간
- C_IMPORT_IMAGE_FILENAME
  - 읽어올 이미지 파일
- C_EXPORT_IMAGE_FILENAME
  - 결과물을 출력할 이미지 파일
- C_EXPORT_IMAGE_WIDTH, C_EXPORT_IMAGE_HEIGHT
  - 출력 이미지의 가로(width) 세로(height) 크기
- C_EXPORT_IMAGE_TOTAL_PIXELS
  - 출력 이미지의 총 픽셀 개수
- C_NUMBER_OF_DIVISION
  - 작업 분할 갯수

코드 설명 (main.c) : 변수

- **BLACKHOLE_LAB *blackhole_lab = NULL;**
- **RESULT *result = NULL;**
- **int number_of_results;**
- **RESULT *collection = NULL;**
- **int x_start, x_end;**
- **int y_start, y_end;**
- **int my_rank, num_procs;**
- **int i, j;**



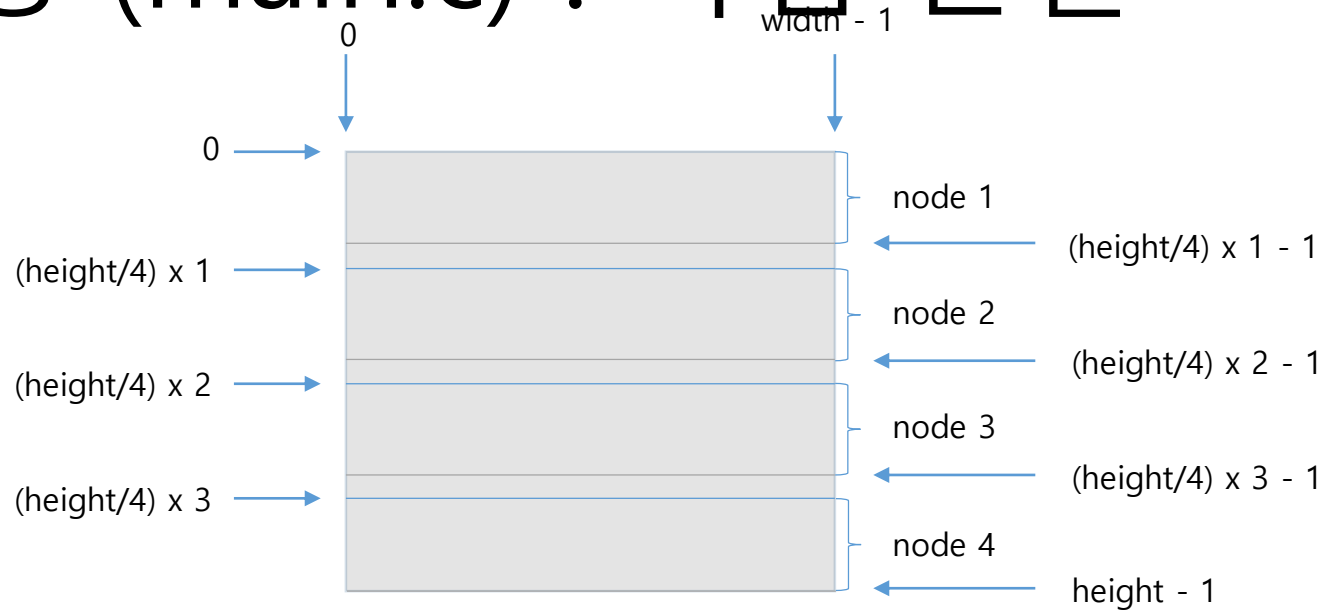
- blackhole_lab
  - 시뮬레이션이 이루어지는 구조체
- result
  - 시뮬레이션 결과물을 저장하는 배열
- number_of_result
  - 결과물의 개수 = 빛다발의 개수
- collection
  - 결과물을 모으는 배열
- x_start, x_end, y_start, y_end
  - 시뮬레이션 범위
- my_rank
  - 작업 분할 번호
- num_procs
  - 작업 분할 갯수

코드 설명 (main.c) : 작업 분할

- **collection = (RESULT*)calloc(C_EXPORT_IMAGE_TOTAL_PIXELS, sizeof(RESULT));**
- **num_procs = C_NUMBER_OF_DIVISION;**
- **x_start = 0;**
- **x_end = C_EXPORT_IMAGE_WIDTH - 1;**
- **for (my_rank = 0, i = 0; my_rank < num_procs; my_rank++) {**
- **y_start = (C_EXPORT_IMAGE_HEIGHT / num_procs) * my_rank;**
- **y_end = (C_EXPORT_IMAGE_HEIGHT / num_procs) * (my_rank + 1) – 1;**
- **.....**
- **}**
- **BLACKHOLE_LAB_make_vision(collection, C_IMPORT_IMAGE_FILENAME, C_EXPORT_IMAGE_FILENAME, C_EXPORT_IMAGE_WIDTH, C_EXPORT_IMAGE_HEIGHT);**
- **free(collection);**

- collection : calloc으로 C_EXPORT_IMAGE_TOTAL_PIXELS 만큼의 메모리를 잡았다가 최종적으로는 free로 해제
- x_start와 x_end는 모든 작업에 대해 고정
- y_start와 y_end를 지정하여 시뮬레이션 영역 분할
- my_rank가 0부터 num_procs - 1까지 증가하며 분할된 작업을 하나하나 실행
- BLAKCHOLE_LAB_make_vision이 collection에 담긴 결과물을 이미지로 출력

코드 설명 (main.c) : 작업 분할



노드	시작 인덱스	끝 인덱스	점 개수
node 1	0	(height/4) x 1 – 1	height/4
node 2	(height/4) x 1	(height/4) x 2 – 1	height/4
node 3	(height/4) x 2	(height/4) x 3 – 1	height/4
node 4	(height/4) x 3	height – 1	height/4

코드 설명 (main.c) : 시뮬레이션 실행

```
blackhole_lab = BLACKHOLE_LAB_create(
        C_EXPORT_IMAGE_WIDTH, x_start, x_end,
        C_EXPORT_IMAGE_HEIGHT, y_start, y_end,
        C_VISUAL_ANGLE, C_TIME_STEP, C_TIME_INTERVAL
);

if (NULL == blackhole_lab) {
        BLACKHOLE_LAB_destroy(blackhole_lab);
        fprintf(stderr, "Creating BLACKHOLE_LAB is Failed.₩n");
        return -1;
}

BLACKHOLE_LAB_start_experiment(blackhole_lab);
```

- BLACKHOLE_LAB_crate로 블랙홀 실험실을 세팅함.
- 블랙홀 실험실 생성에 실패하면 에러메시지 출력 후 종료.
- BLACKHOLE_LAB_start_experiment로 블랙홀 실험실 가동.

코드 설명 (main.c) : 결과물 모으기

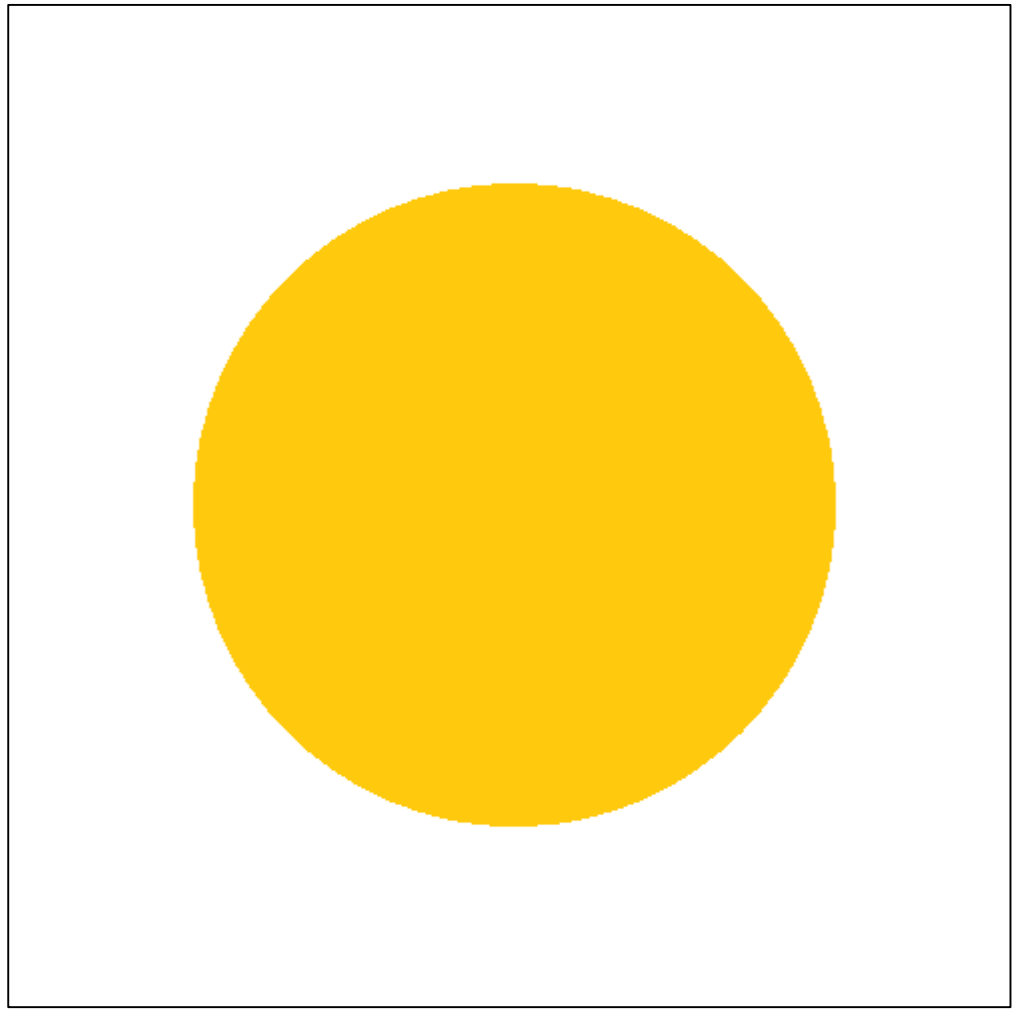
- **number_of_results = C_EXPORT_IMAGE_WIDTH * C_EXPORT_IMAGE_HEIGHT / num_procs;**
- **result = (RESULT*)calloc(number_of_results, sizeof(RESULT));**
- **BLACKHOLE_LAB_dump_results(blackhole_lab, result);**
- **BLACKHOLE_LAB_destroy(blackhole_lab);**
- **for (j = 0; j < number_of_results; j++, i++) {**
- **collection[i] = result[j];**
- **}**
- **free(result);**

- result : calloc으로 number_of_results 만큼의 메모리를 잡았다가 최종적으로는 free로 해제
- BLACKHOLE_LAB_dump_results로 result에 결과물을 쏟아냄
- 결과물을 쏟아낸 블랙홀 실험실은 더 이상 필요가 없으므로 BLACKHOLE_LAB_destro로 소멸시킴.
- result의 결과들을 collection에 차곡차곡 모음.

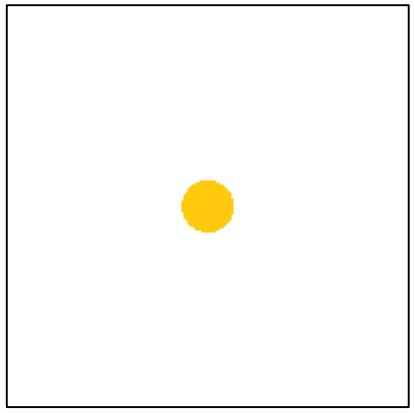
컴파일 및 실행 방법

- 소스코드 압축 해제
  - `mkdir blackhole_lab`
  - `cp source.tar blackhole_lab`
  - `cd blackhole_lab`
  - `tar -xf source.tar`
- libpng 설치
  - `sudo apt-get install libpng-dev`
- 컴파일
  - `make`
- 재컴파일
  - `make clean`
  - `make`
- 실행
  - `./blackhole_lab`
- 병렬화 실행
  - `mpiexec -np 4 -hostfile hostfile ./blackhole_lab`

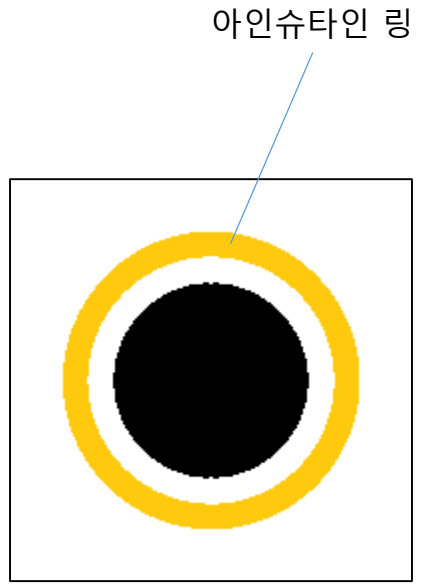
시뮬레이션 결과



input.png (원본 이미지)



블랙홀이 없는 경우의 상



블랙홀이 있는 경우의 상

과제

- 과제 1 : input.png 를 자신이 원하는 이미지로 하여 프로그램을 실행하고 자신만의 블랙홀 이미지를 만드시오.
  - png 형식의 이미지만 가능
- 과제 2 : node 4개를 사용하여 계산하는 병렬화 코드를 작성하시오.
  - main.c 만 수정
  - 데이터를 한곳으로 모으는 작업은 소스 코드 내에 주어진 gather_results 함수를 적절히 활용.
- 과제 3 : 작업을 오른쪽 그림과 같이 분할하시오.
- 제출
  - 제출 : hpcyouthcamp@kisti.re.kr
  - 제출파일 : 과제1의 이미지 파일, 과제2의 main.c, 과제3의 main.c를 압축하여 메일로 첨부
  - 기한 : 목요일 정오