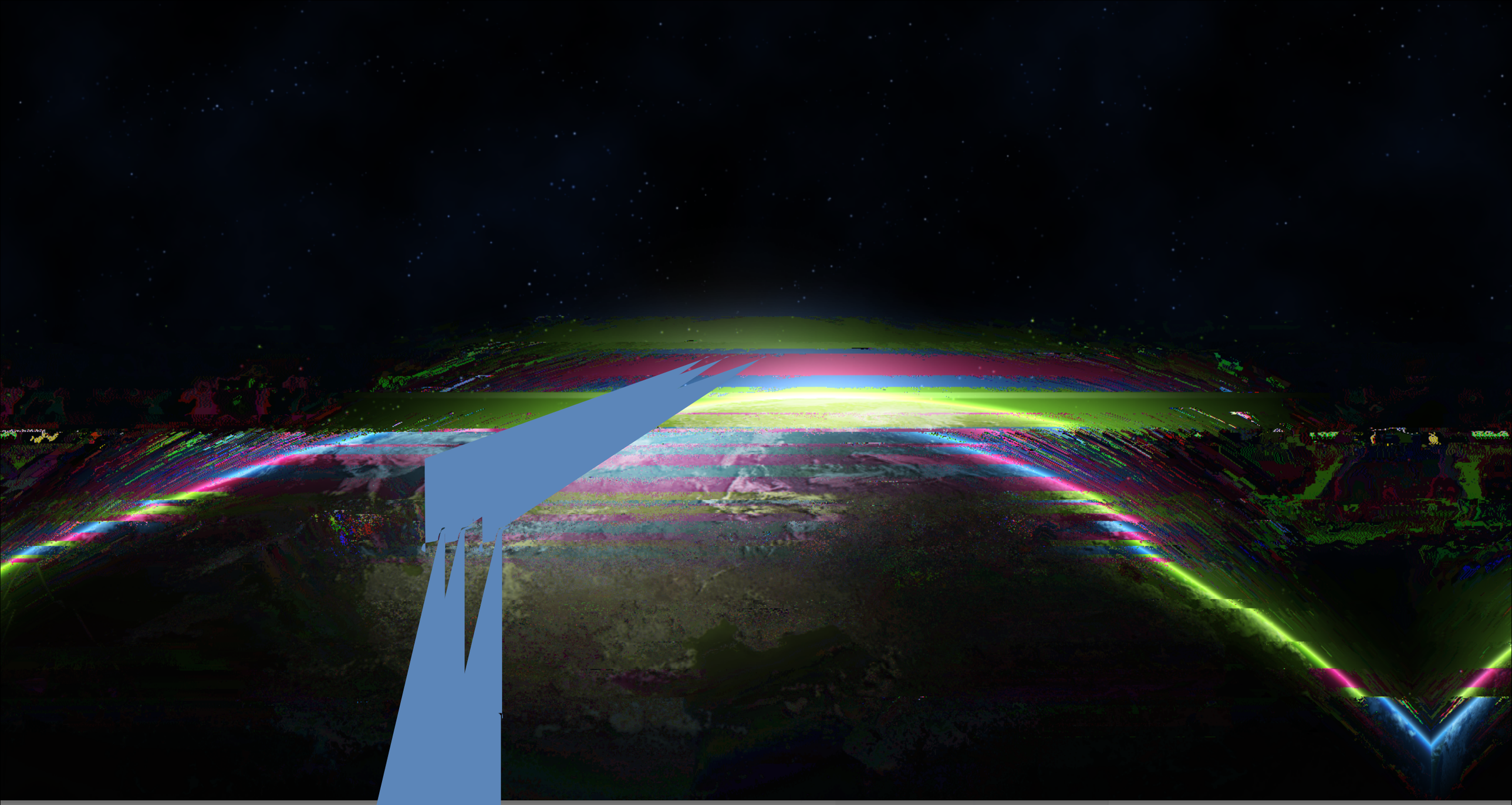
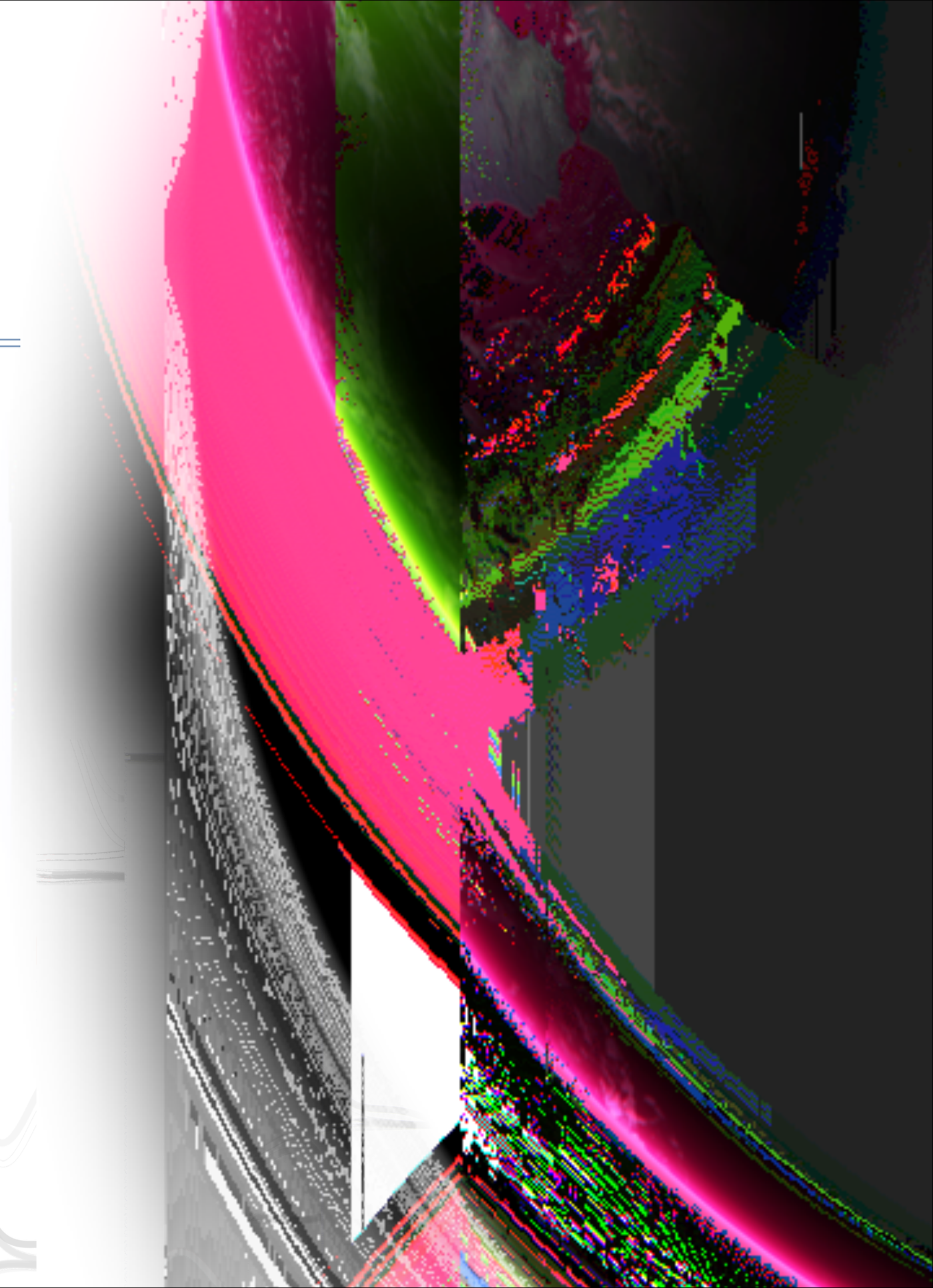


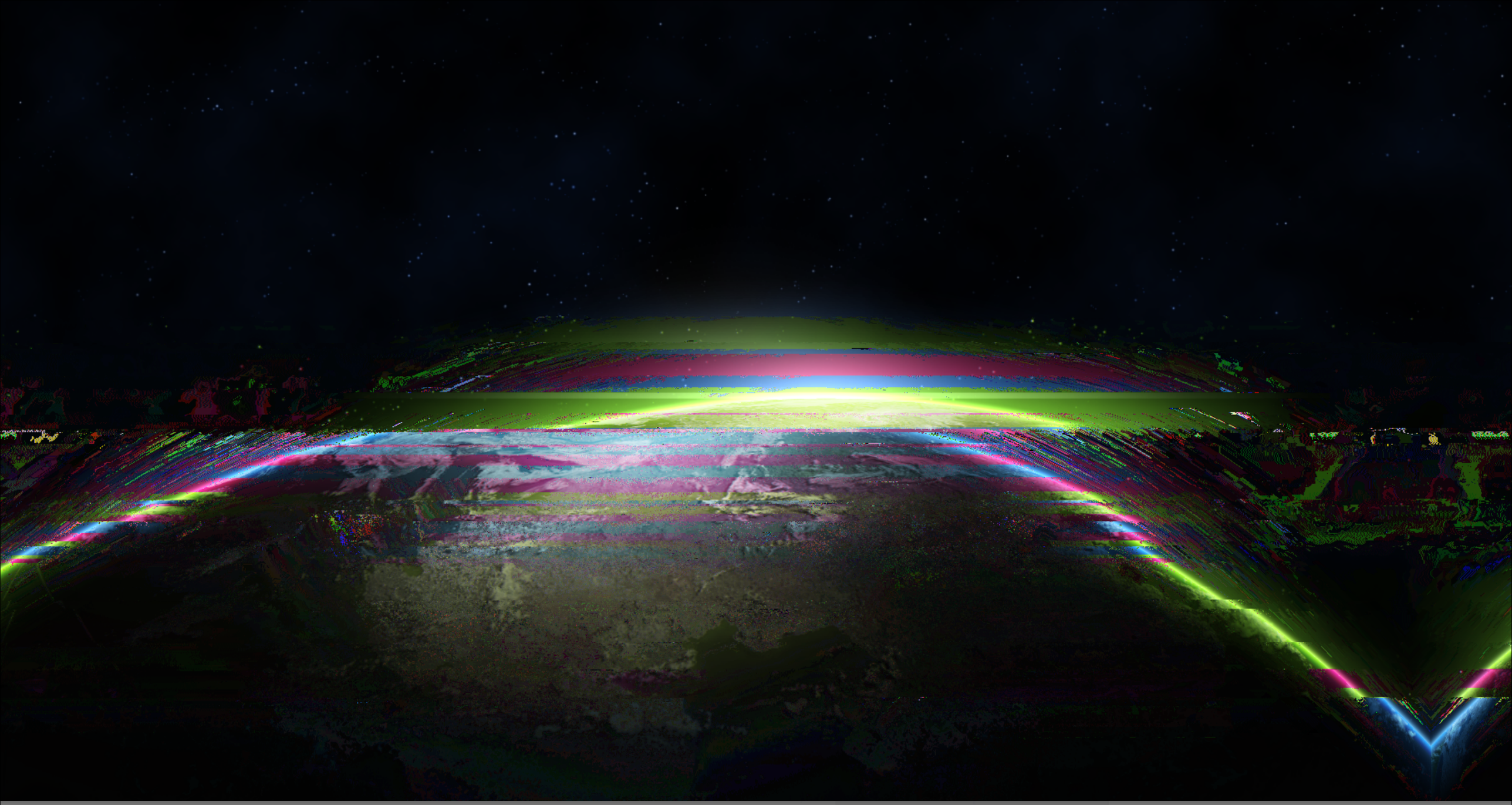
Teachin CE408 at Scale

Abd Da a

Overview

- IMPACT
- Hi
- A chi ec e
- S e da a
- Le
- F e





Quick Demo

IMPACT

- N hi g i i a
  - W f CU O e CL/O e ACC/
- P ib he i ib e ec i g
  - Th a d e h d ed f h he i e
- P d c i h i e e a i

Objective

- C ea e a g a i g e i e f he C e a c e
- A e e de e ide f he e i e
- N be ied i C e a (a ffe i f e ch)

Previous System

- Ma e i e face i e
- Had b e i h ca i g
- G adi g a d e f i e
- Did ha e ee e ie i g

New Syste

- I a g adi g a d
  b i i bac C
- Pee e ie i e
- Sca i g a
- Ma i g a e i

How do you grade 4000 Programs?

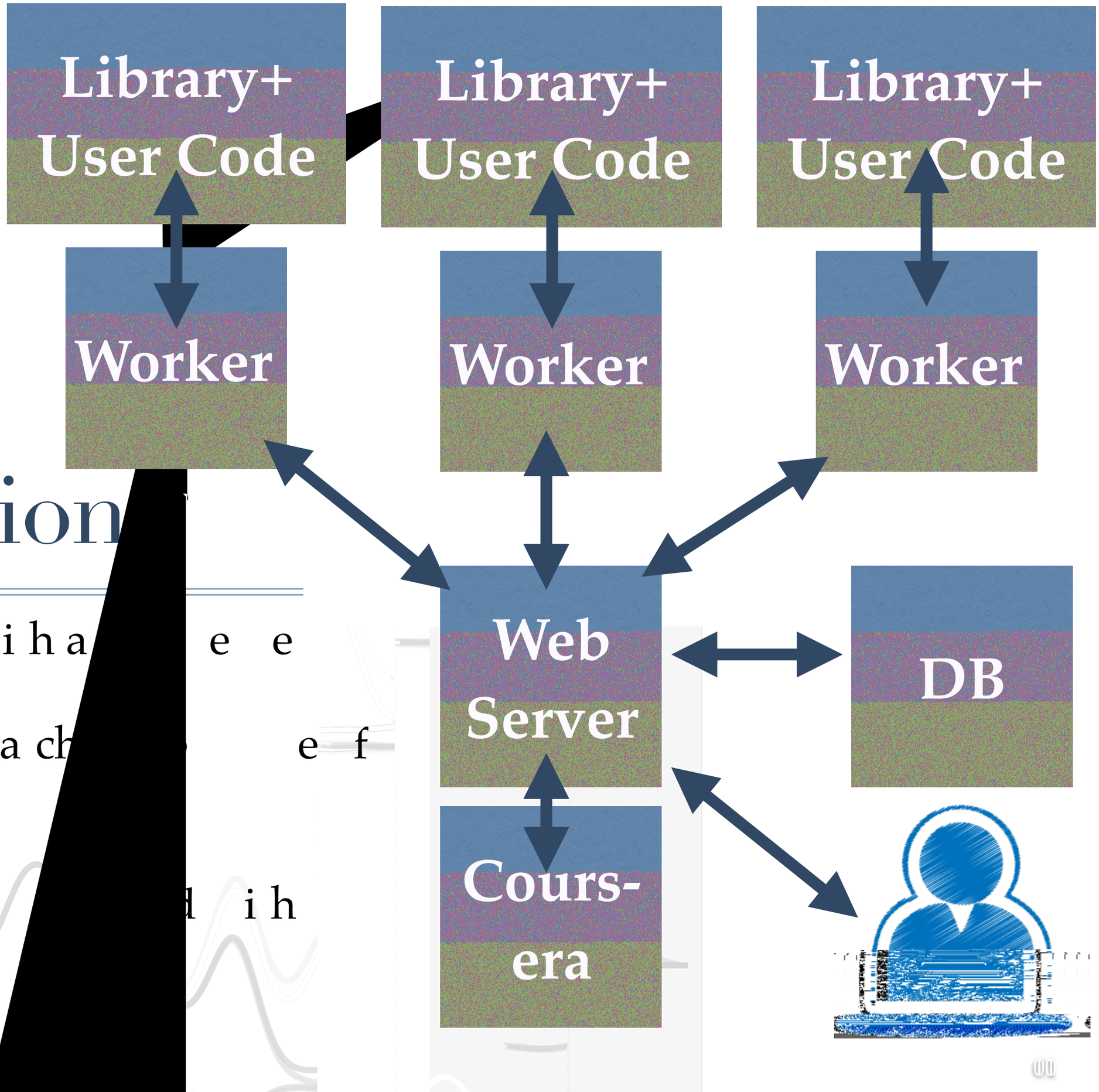
Y d .

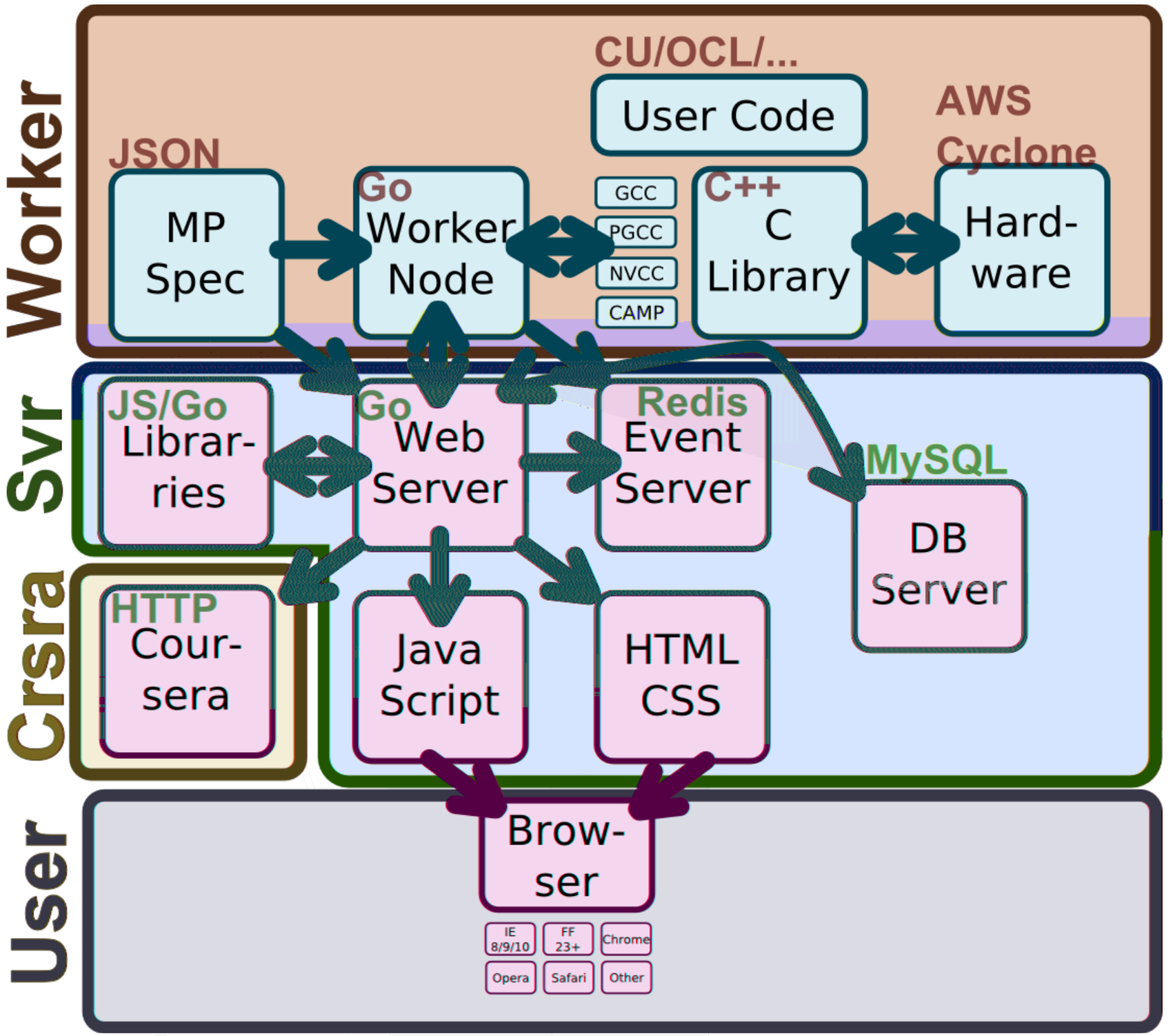
Architecture

Life of a Program Submission

Life of a Program Submission

- U e i e ac i h a e e
- Web e e di a ch e f a e
- G ade ge c d i h C e a





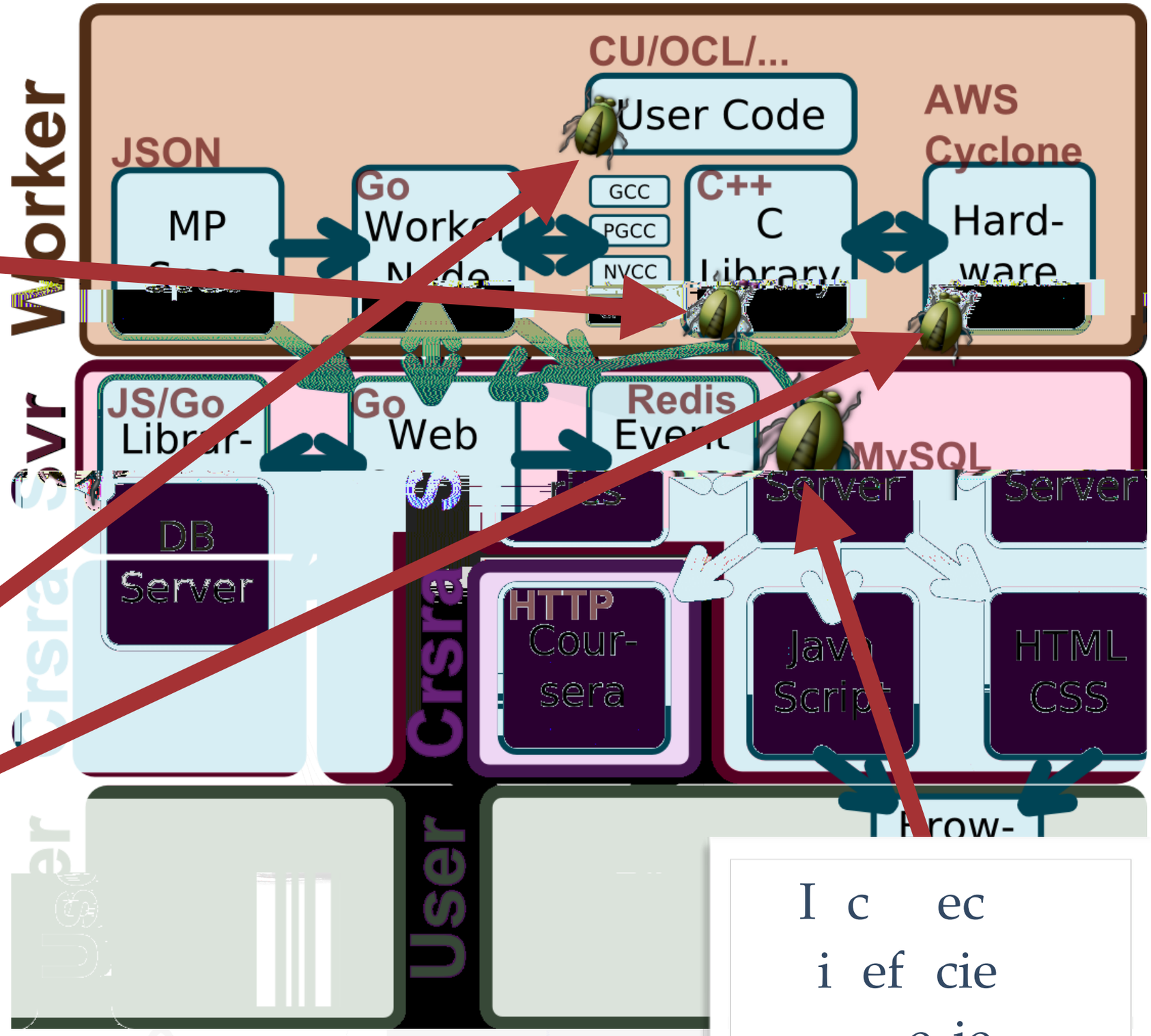
Detailed Architecture

Source of Bugs

Made i b ic, g e he f de

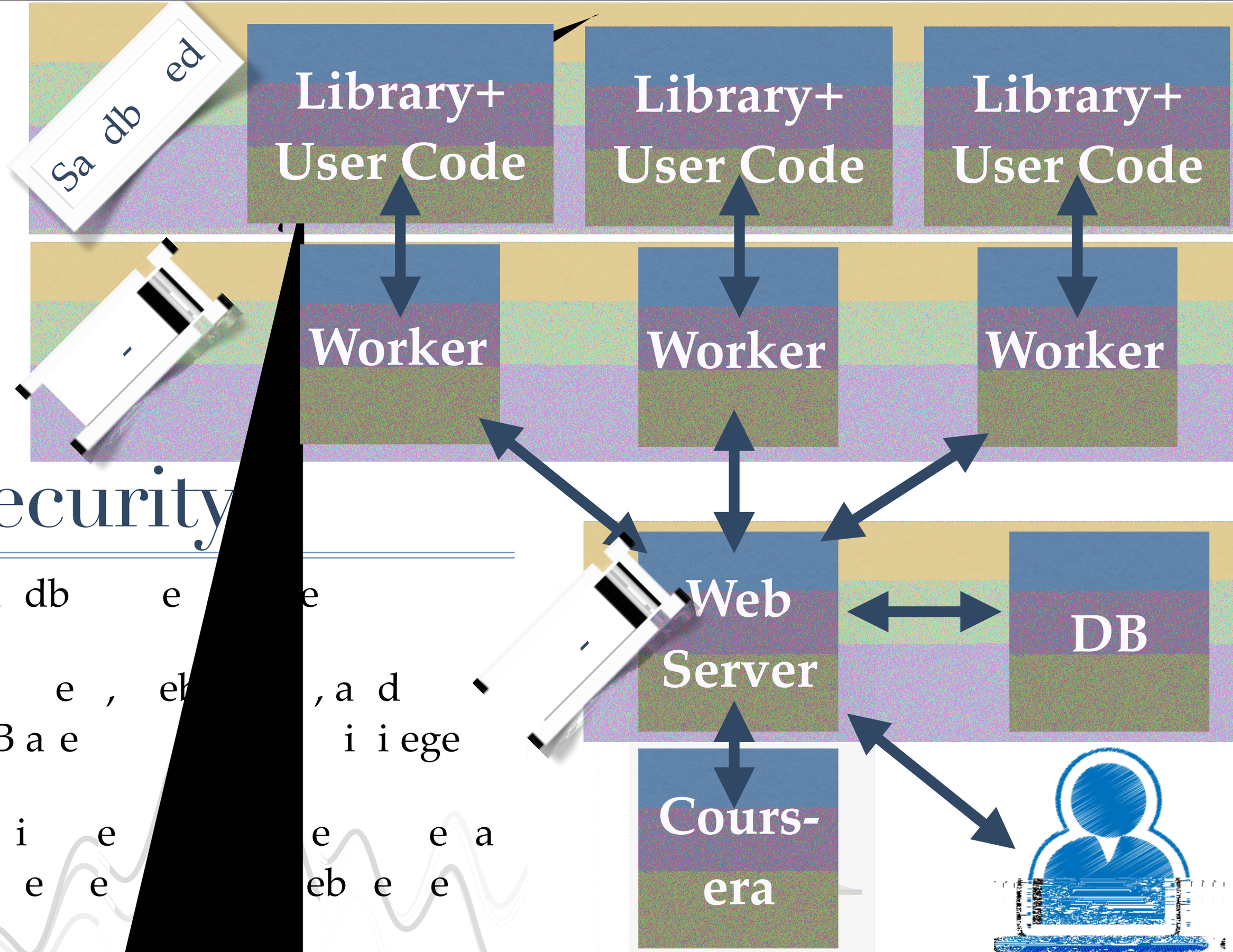
C e a ed a d ce f a b e (ha e C c e a d AWS)

I c ec i ef cie e ie



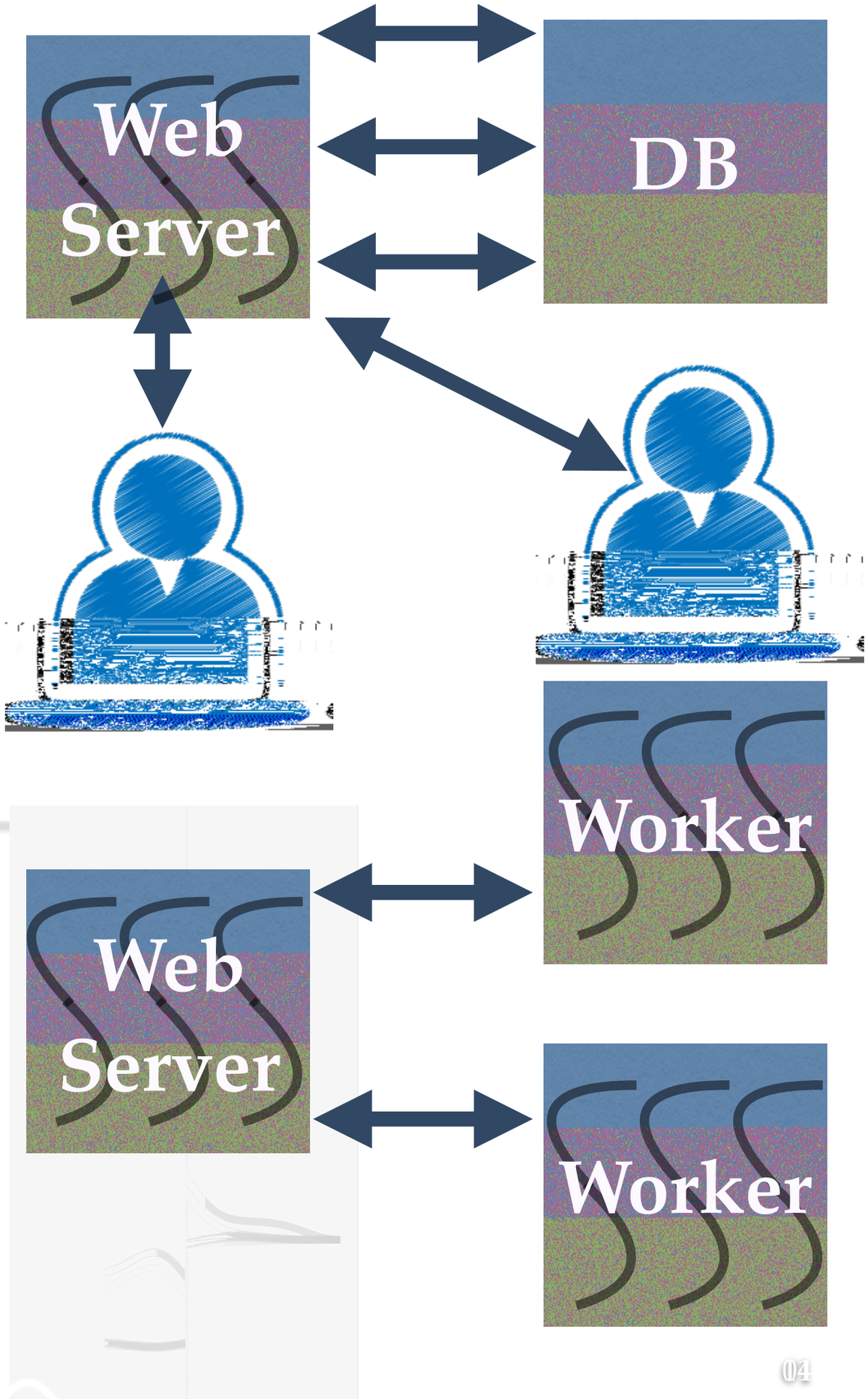
Security

- Sa db e e
- W e , eb , a d DB a e i ege
- A i e e e a 80 e e eb e e

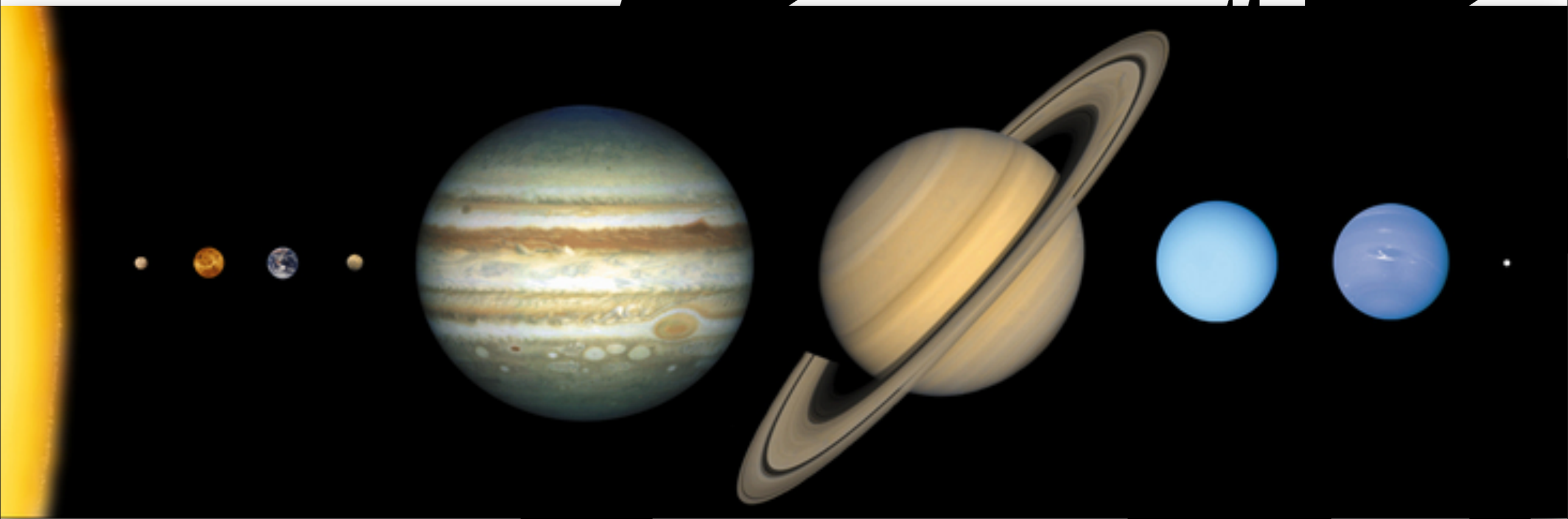


How to Scale?

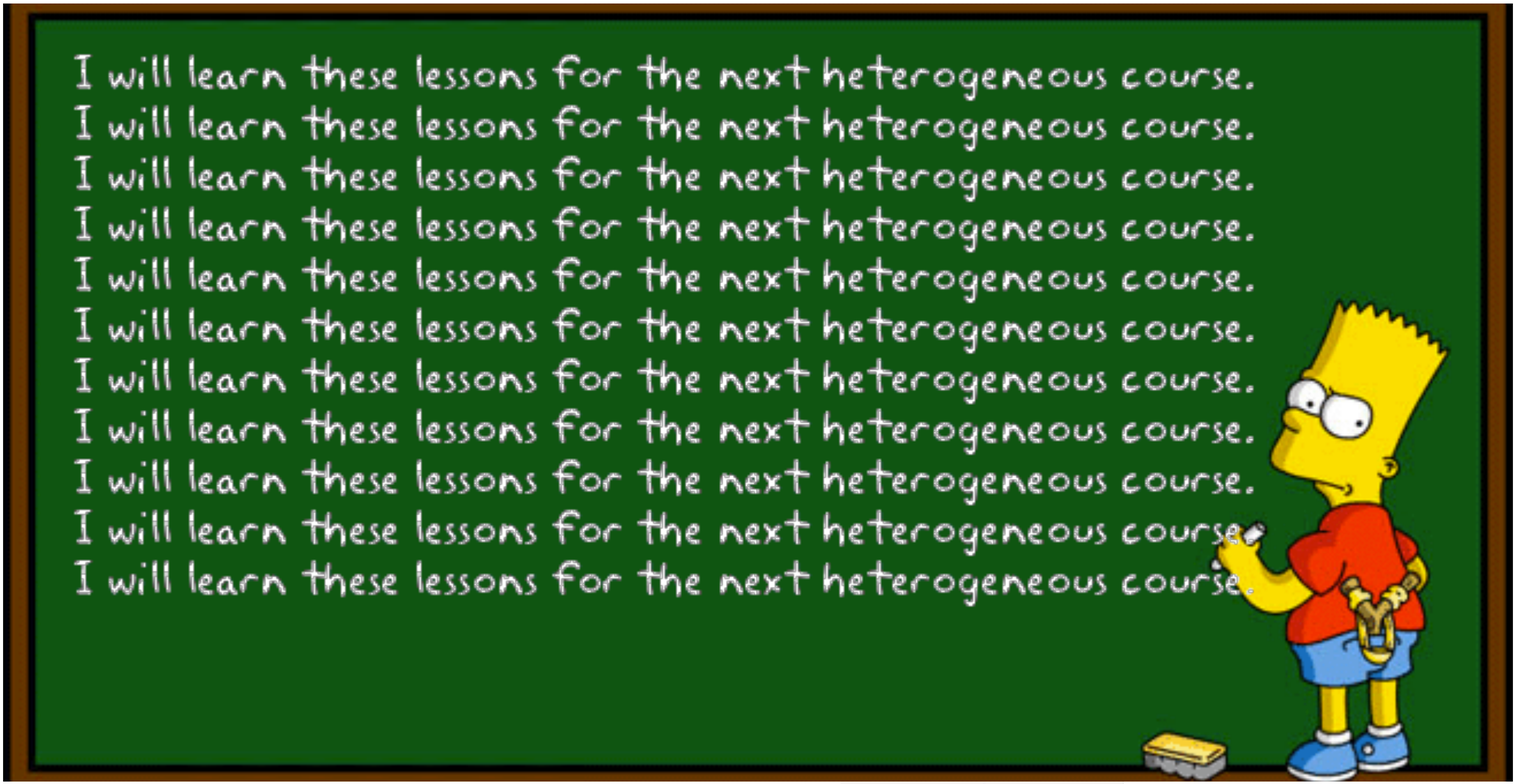
- Each e e i a igh eigh ead (h a d f he e)
- The igh eigh h ge a ed *n* h ead (*n* bei g h be f c e)
- C ea e a c ec i i h he da aba e e e
- M f e a i e a ch
- Ma e ca c a e a be f e
- W e i a diffe e GPU if a ai



Scale



- Wha f 2 a f 100
- Wha f 1 e a l gge a he a e i e
- Wha f e da 1 a 00 e e da

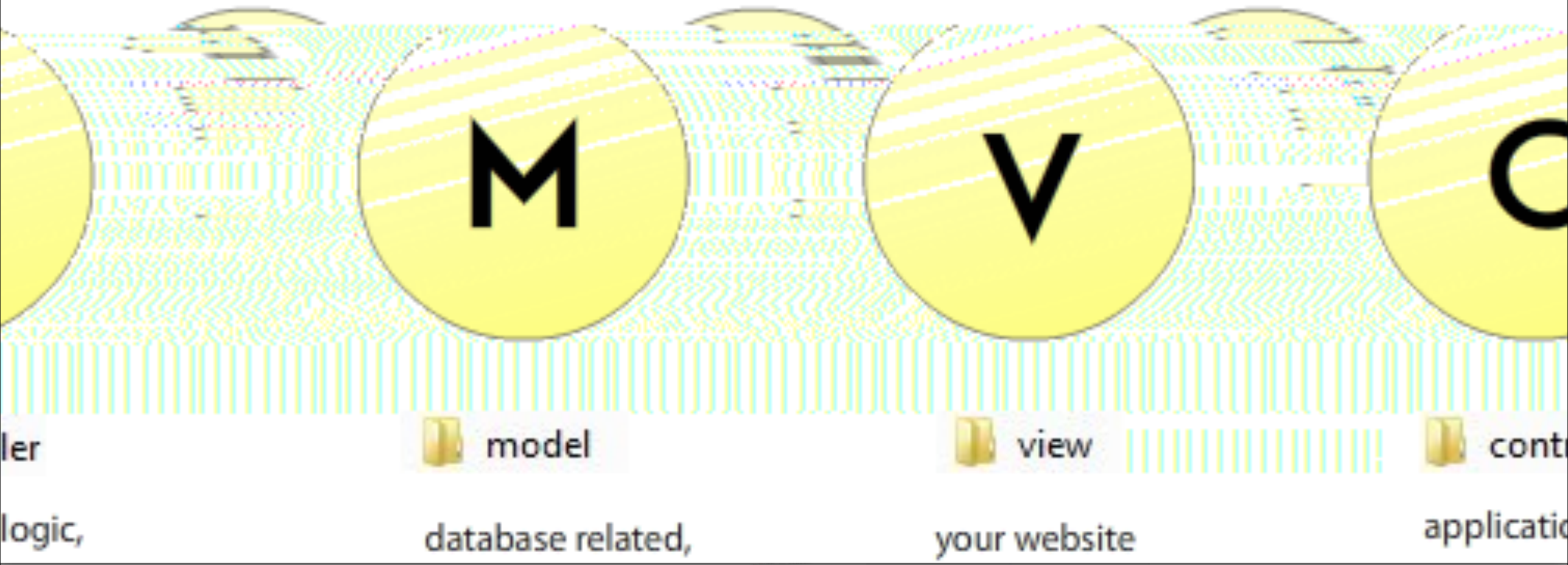


Lessons Learned (Implementation)

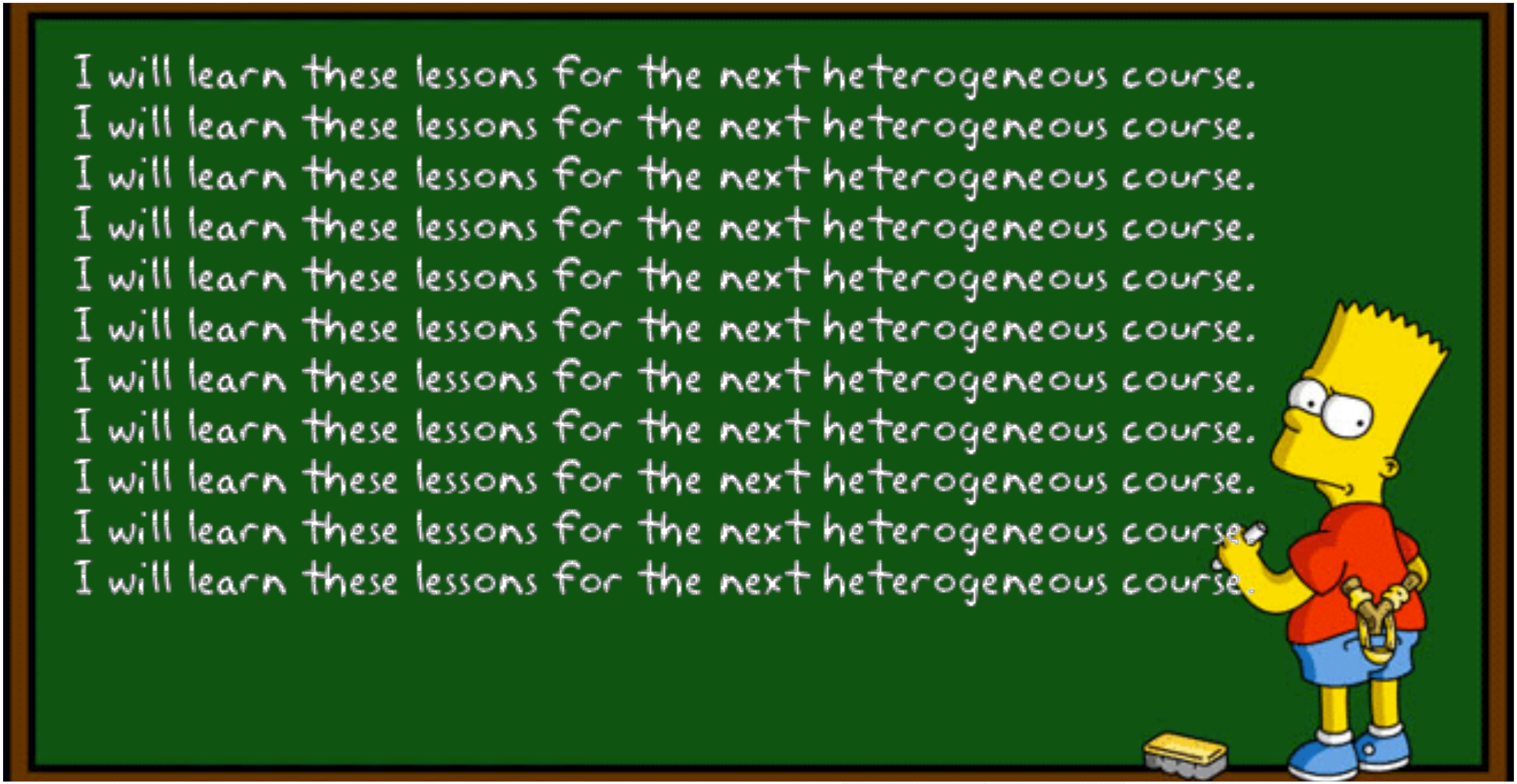
Abstraction is not a Good Thing

- A he a f he c e, e e e i g a ib a i if da aba e e ie i did i if he , b ge e a ed c e e ie ha e e ib e deb g

Abstraction is a Great Thing



- Whe e e aced he ib a , had e ace he de c de i he a ica i



Lessons Learned (Human)

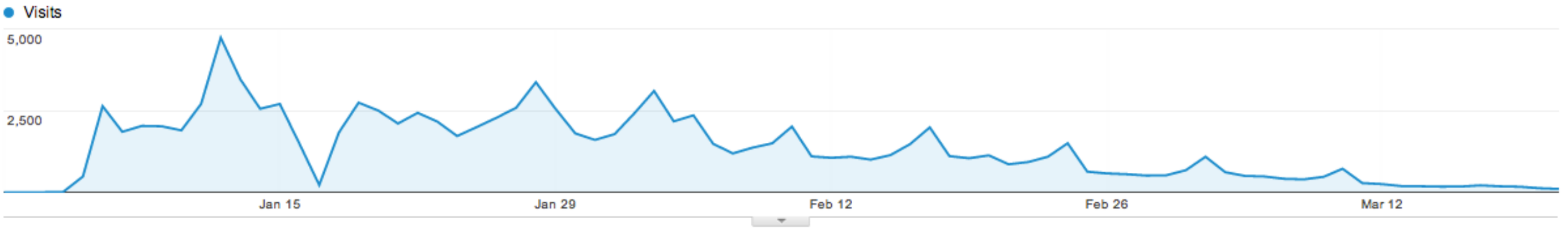
Lessons Learned

- Nega i e c i ici ea de i i e e
  - Y i fee di c aged a f he e f he da
- Pe e d ead d c e ea ch he f
  - Y i a e he a e a d e d e agai
- Pe e i e d hei he
  - The i fee i e e he

Data

Data Collected

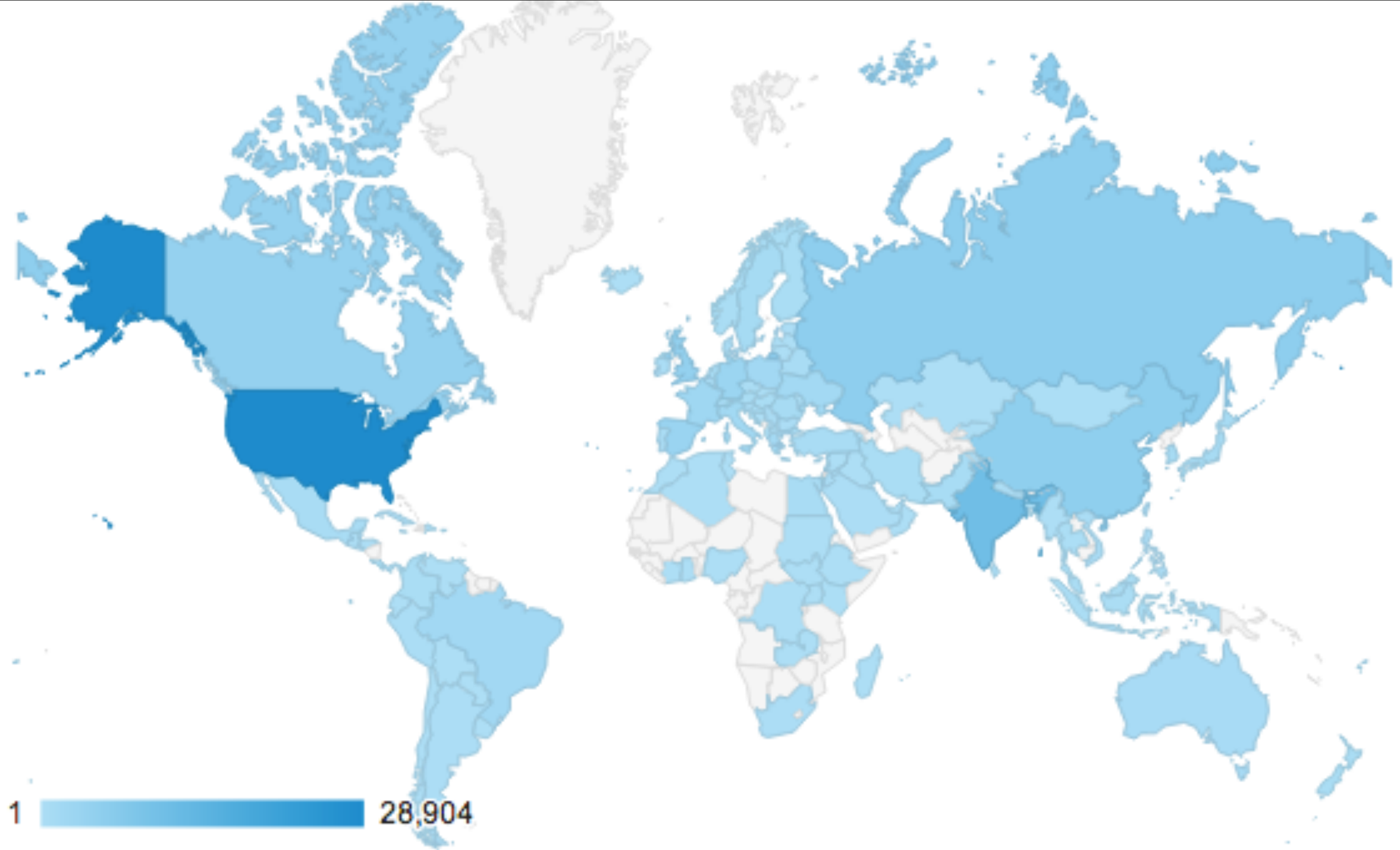
- G g e A a ic
- A 13Gb da aba e c ai i g
  - 2 Mi i g a e i i
  - 700 h a d g a
    - R i e/C i a i i e/e
  - 6 h a d g aded g a
- A 7Gb E e da aba e c ai i g
  - CPU/GPU i f a i
  - Wha age e Vi i ed



Visits	Unique Visitors	Pageviews	Pages / Visit
106,427	17,865	998,517	9.38

Avg. Visit Duration	Bounce Rate	% New Visits
00:14:20	20.54%	16.60%

Thousands of Visitors



Users from All over the World

Visits
106,427

Unique Visitors
17,865

Pageviews
998,517

Pages / Visit
9.38

Avg. Visit Duration
00:14:20

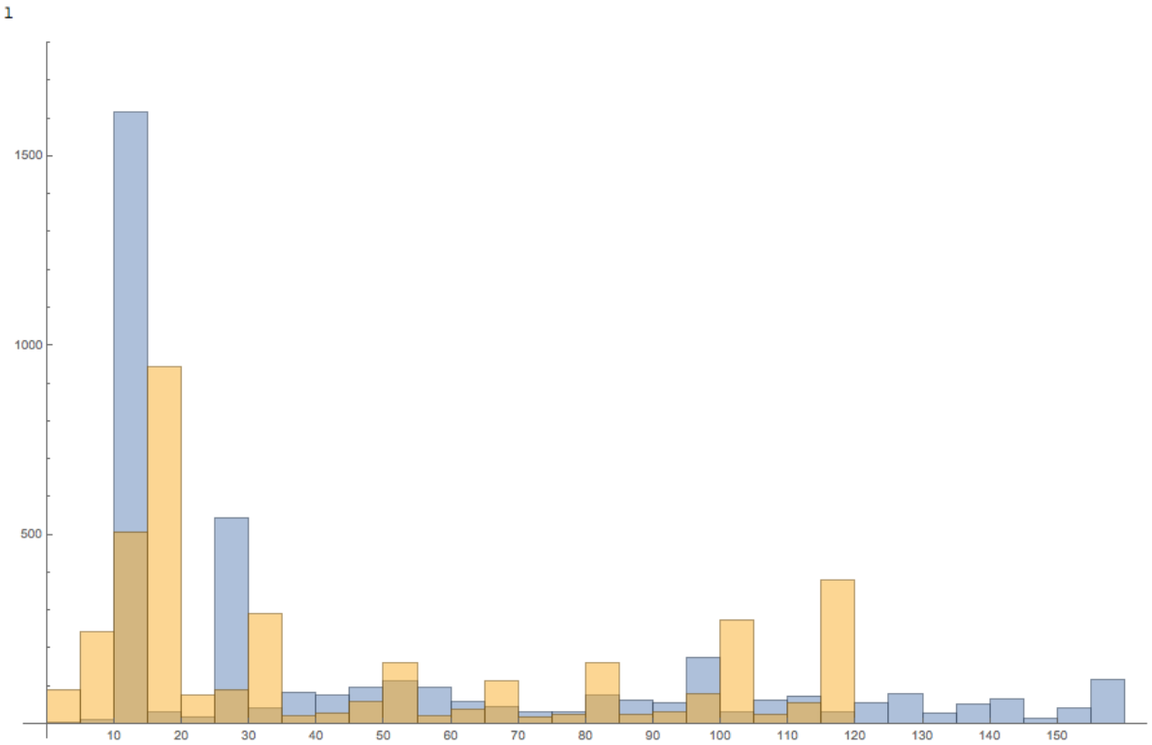
Bounce Rate
20.54%

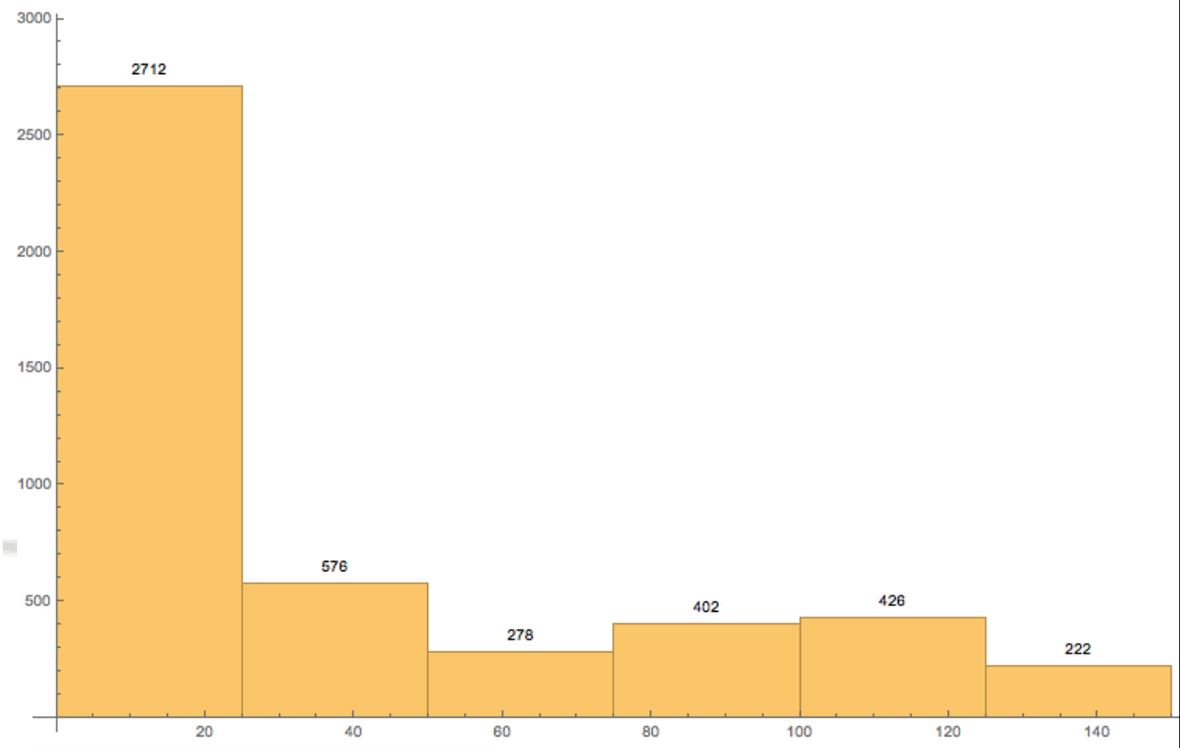
% New Visits
16.60%

People Spend Time on the Site

Grades

Most People Passed





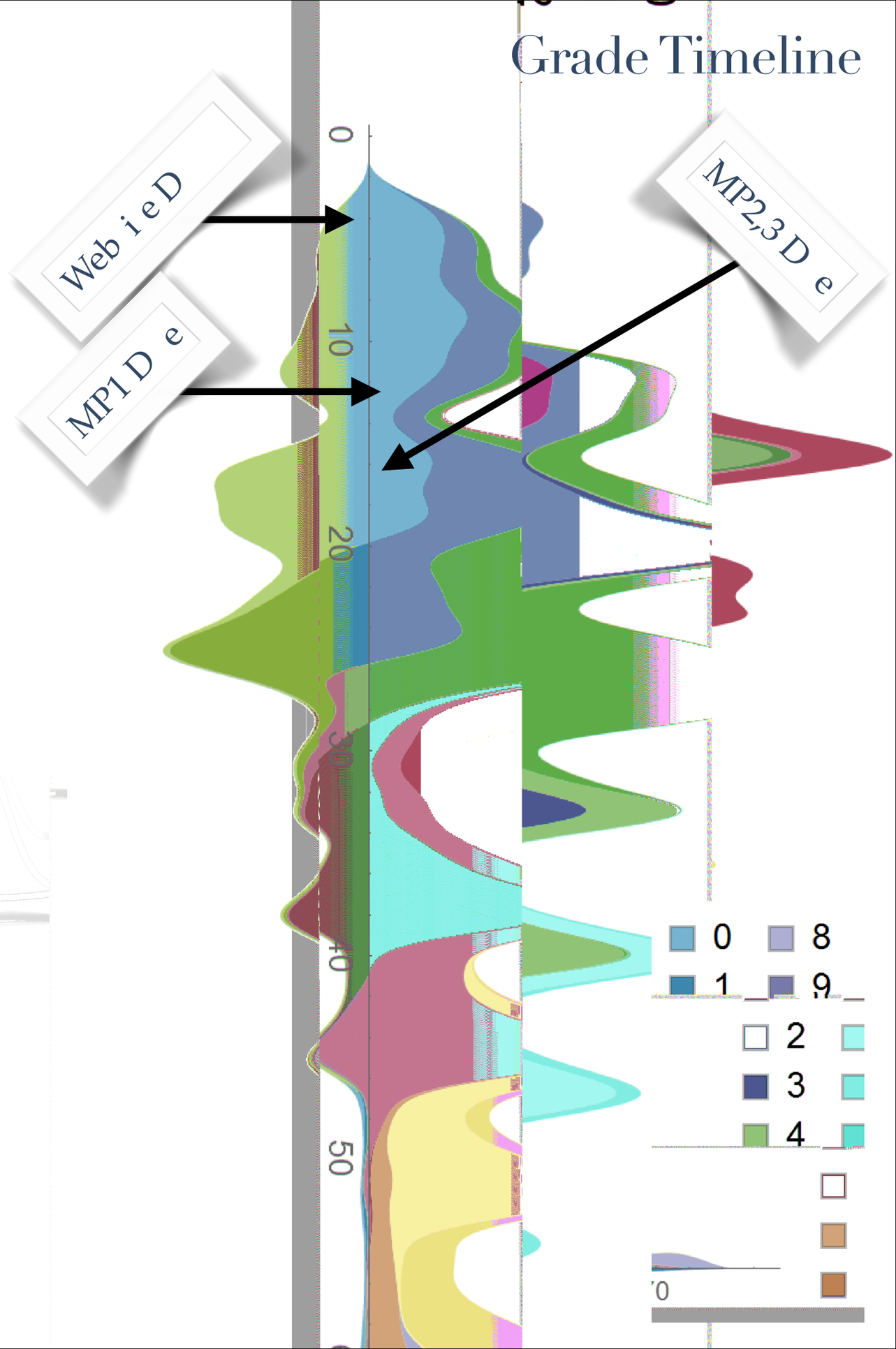
- 939 ac a a ed he c e XXXX hi i c ec FIXME
- 989 g e 80%
- 648 g e 100%

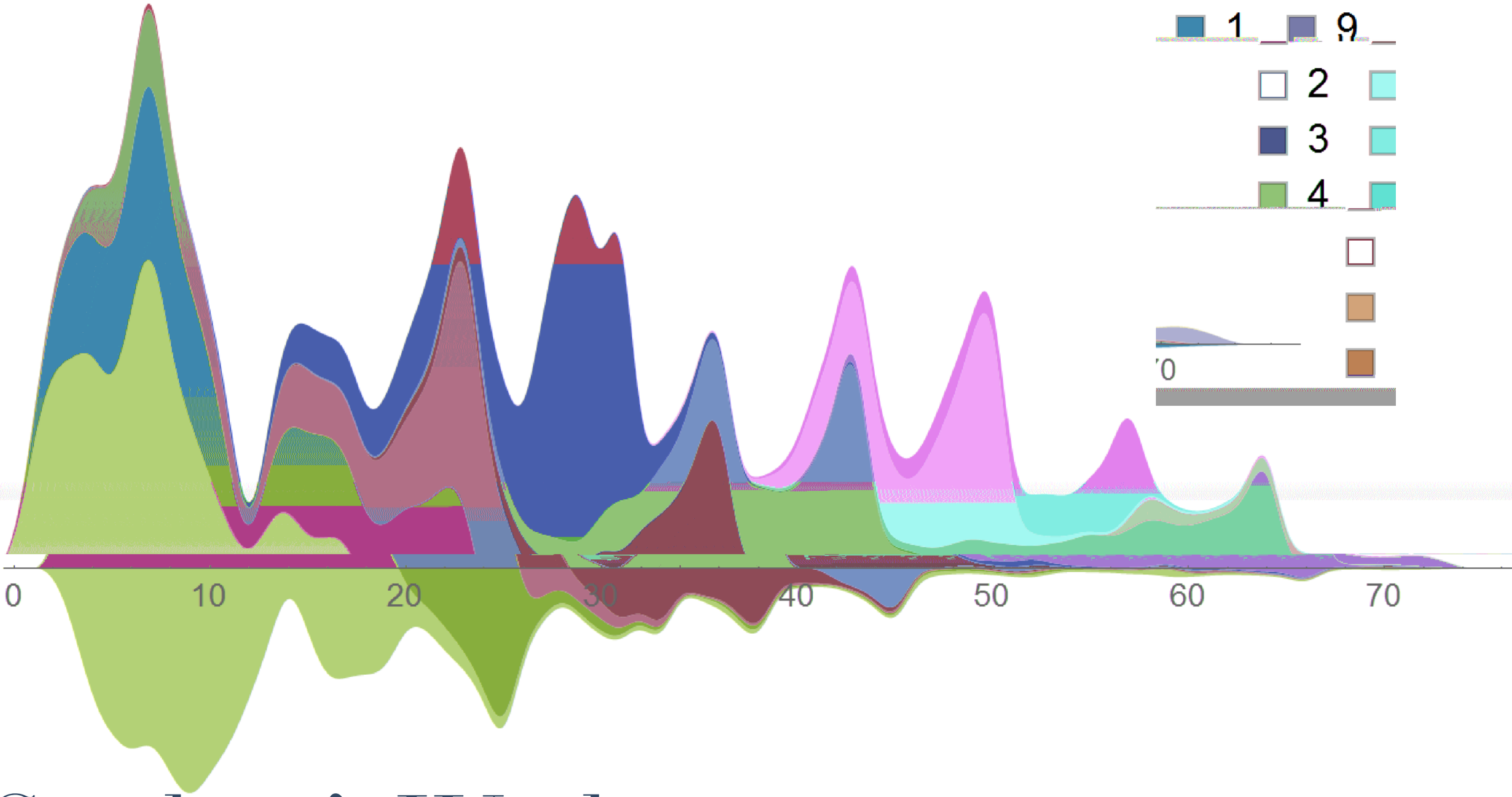
Some Data Insights

Date

People Submit at the Last Minute

- We ade a c e MP a ai ab e da 1
- E e e ai ed i he a i e
- E e MP ha 2 b e f b i i g he c de a d a he f he ee e ie

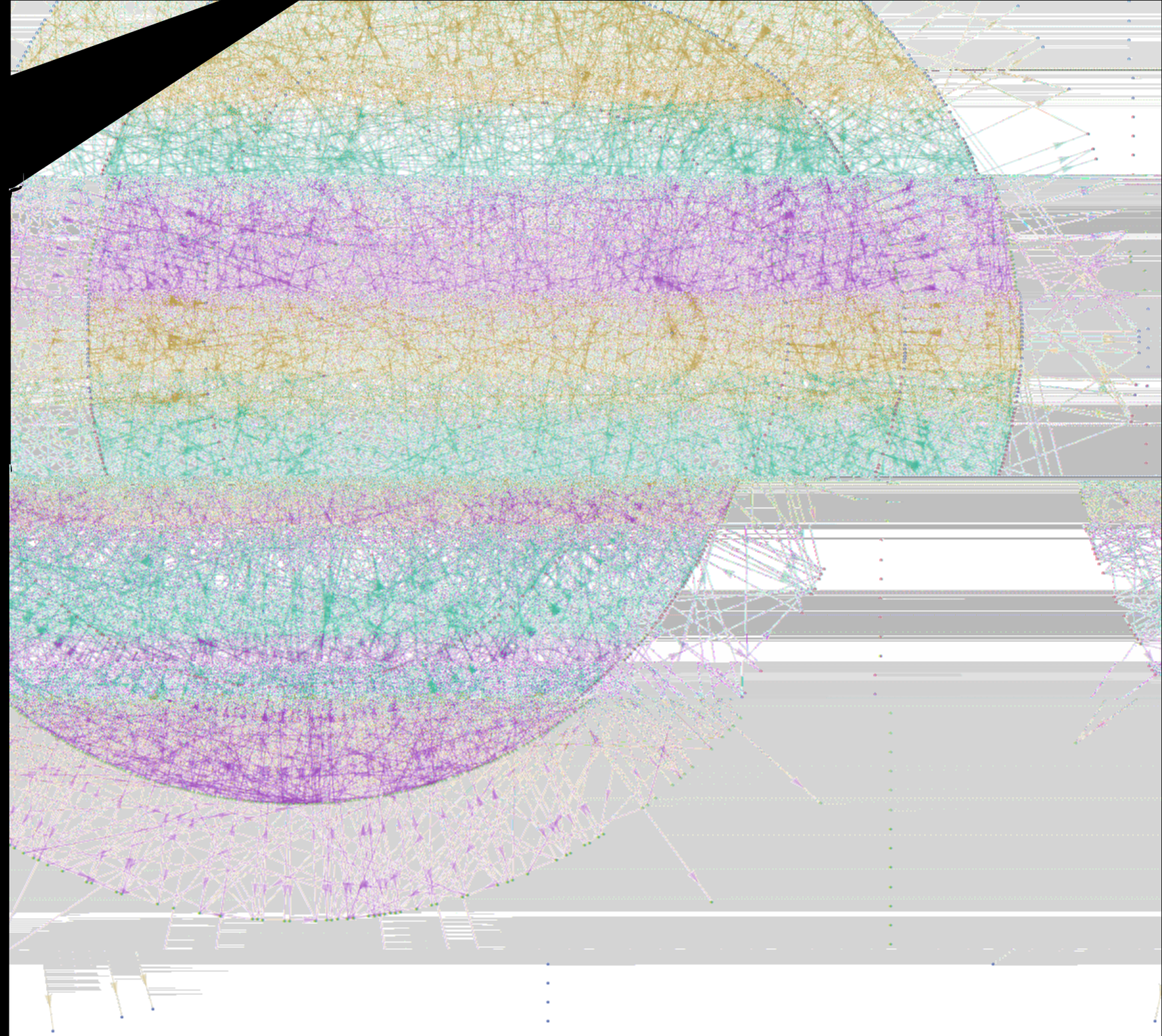




Student's Work

P g a a e i e i e

- S e e ed
  gi e ecei e
  feedbac
- S e e e e
  feedbac b d
  gi e a
- S e e e
  feedbac b
  ecei e a



Peer R Relies on Participation

Data Analysis Opportunities

SGEMM Implementation

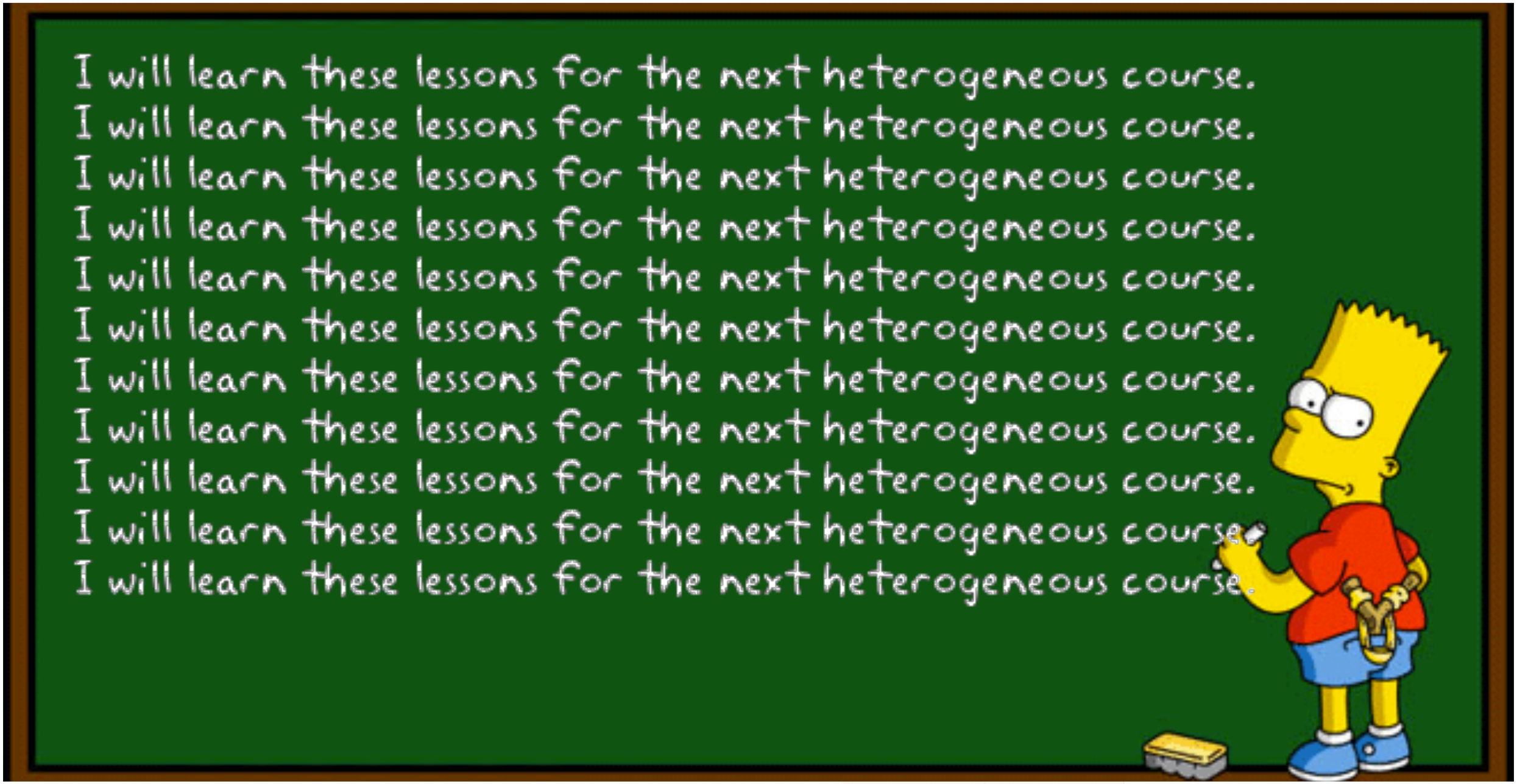
```
#define BLOCK_SIZE 16

__global__ void MatMulKernel(float *DA, float *DB, float *DC, int Ah, int Aw,
                             int AwTiles, int Bh, int Bw) {
  // Block row and column
  int blockRow = blockIdx.y;
  int blockCol = blockIdx.x;

  // Thread row and column within Csub
  int row = threadIdx.y;
  int col = threadIdx.x;

  int cellRow = blockRow * BLOCK_SIZE + row;
  int cellCol = blockCol * BLOCK_SIZE + col;

  // Each thread computes one element of Csub
  // by accumulating results into Cvalue
  float Cvalue = 0.0;

// Loop over all the sub-matrices of A and B that are
// required to compute Csub
// Multiply each pair of sub-matrices together
// and accumulate the results

#pragma unroll
  for (int m = 0; m < AwTiles; ++m) {

    // Shared memory used to store Asub and Bsub respectively
    __shared__ float As[BLOCK_SIZE][BLOCK_SIZE];
    __shared__ float Bs[BLOCK_SIZE][BLOCK_SIZE];

    // Load Asub and Bsub from device memory to shared memory
    // Each thread loads one element of each sub-matrix
    int aRow = BLOCK_SIZE * blockRow + row;
    int aCol = BLOCK_SIZE * m + col;

    float aValue = ((aRow < Ah) && (aCol < Aw));
    aValue *= DA[Aw * aRow + aCol];
    As[row][col] = aValue;

    int bRow = BLOCK_SIZE * m + row;
    int bCol = BLOCK_SIZE * blockCol + col;

    float bValue = ((bRow < Bh) && (bCol < Bw));
    bValue *= DB[Bw * bRow + bCol];
    Bs[row][col] = bValue;

    // Synchronize to make sure the sub-matrices are loaded
    // before starting the computation
    __syncthreads();

    // Multiply Asub and Bsub together
    for (int e = 0; e < BLOCK_SIZE; ++e)
      Cvalue += As[row][e] * Bs[e][col];

    // Synchronize to make sure that the preceding
    // computation is done before loading two new
    // sub-matrices of A and B in the next iteration
    __syncthreads();
}

  if (cellRow >= Ah)
    return;
  if (cellCol >= Bw)
    return;

  DC[cellRow * Bw + cellCol] = Cvalue;
}

__global__ void nop() {}


//
//       CITE: Vasily Volkov, UC Berkeley.... Intresting approach to using
//caching a bit differently... had to give it a try...
//
__device__ void saxpy(float a, float *b, float *c) {
  c[0] += a * b[0];
  c[1] += a * b[1];
  c[2] += a * b[2];
  c[3] += a * b[3];
  c[4] += a * b[4];
  c[5] += a * b[5];
  c[6] += a * b[6];
  c[7] += a * b[7];
  c[8] += a * b[8];
  c[9] += a * b[9];
  c[10] += a * b[10];
  c[11] += a * b[11];
  c[12] += a * b[12];
  c[13] += a * b[13];
  c[14] += a * b[14];
  c[15] += a * b[15];
}

__global__ void optimisedDLA(const float *A, int lda, const float *B, int ldb,
                             float *C, int ldc, int k) {
  const int inx = threadIdx.x;
  const int iny = threadIdx.y;
  const int ibx = blockIdx.x * 64;
  const int iby = blockIdx.y * 16;
  const int id = inx + iny * 16;

  A += ibx + id;
  B += inx + __mul24(iby + iny, ldb);
  C += ibx + id + __mul24(iby, ldc);

  const float *Blast = B + k;

  float c[16] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

  do {
    float a[4] = {A[0 * lda], A[1 * lda], A[2 * lda], A[3 * lda]};

    __shared__ float bs[16][17];
    bs[inx][iny] = B[0 * ldb];
    bs[inx][iny + 4] = B[4 * ldb];
    bs[inx][iny + 8] = B[8 * ldb];
    bs[inx][iny + 12] = B[12 * ldb];
    __syncthreads();

    A += 4 * lda;
    saxpy(a[0], &bs[0][0], c);
    a[0] = A[0 * lda];
    saxpy(a[1], &bs[1][0], c);
    a[1] = A[1 * lda];
    saxpy(a[2], &bs[2][0], c);
    a[2] = A[2 * lda];
    saxpy(a[3], &bs[3][0], c);
    a[3] = A[3 * lda];

    A += 4 * lda;

    saxpy(a[0], &bs[4][0], c);
    a[0] = A[0 * lda];
    saxpy(a[1], &bs[5][0], c);
    a[1] = A[1 * lda];
    saxpy(a[2], &bs[6][0], c);
    a[2] = A[2 * lda];
    saxpy(a[3], &bs[7][0], c);
    a[3] = A[3 * lda];

    A += 4 * lda;
    saxpy(a[0], &bs[8][0], c);
    a[0] = A[0 * lda];
    saxpy(a[1], &bs[9][0], c);
    a[1] = A[1 * lda];
    saxpy(a[2], &bs[10][0], c);

    a[2] = A[2 * lda];
    saxpy(a[3], &bs[11][0], c);
    a[3] = A[3 * lda];

    A += 4 * lda;
    saxpy(a[0], &bs[12][0], c);
    saxpy(a[1], &bs[13][0], c);
    saxpy(a[2], &bs[14][0], c);
    saxpy(a[3], &bs[15][0], c);

    B += 16;
    __syncthreads();
  } while (B < Blast);

  for (int i = 0; i < 16; i++, C += ldc)
    C[0] = c[i];
}
….
```

Analysis Opportunities

- Wha e a e c ?
  - Ca e de ec h e a d gi e feedbac
- Wha i i a i a e be e cia ?
  - Which did e ha e he b e i
- Wha ca e GPU c a h?
  - Ca e a id h e?
- Did e e agia i e?
- Wha d e he ee e ie e ?

Source of Data

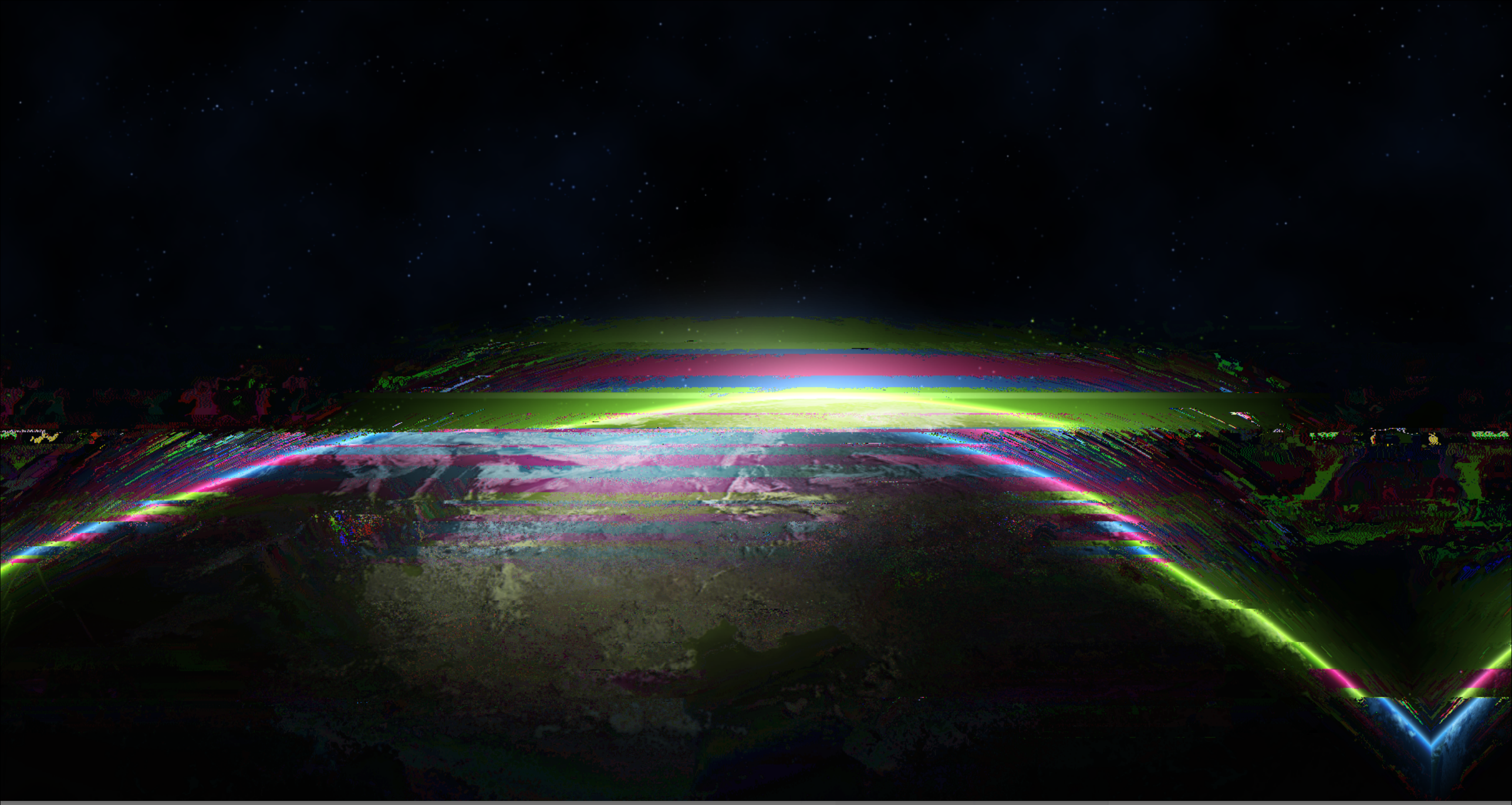
- Thi i a big da a b e
- Diffe e a a i ca be e f ed diffe e a
- P g a a a i g a
- P e a a i ec ded GPU e d a
- NLP e i a d ee e ie



What would be different?

Lessons Learned

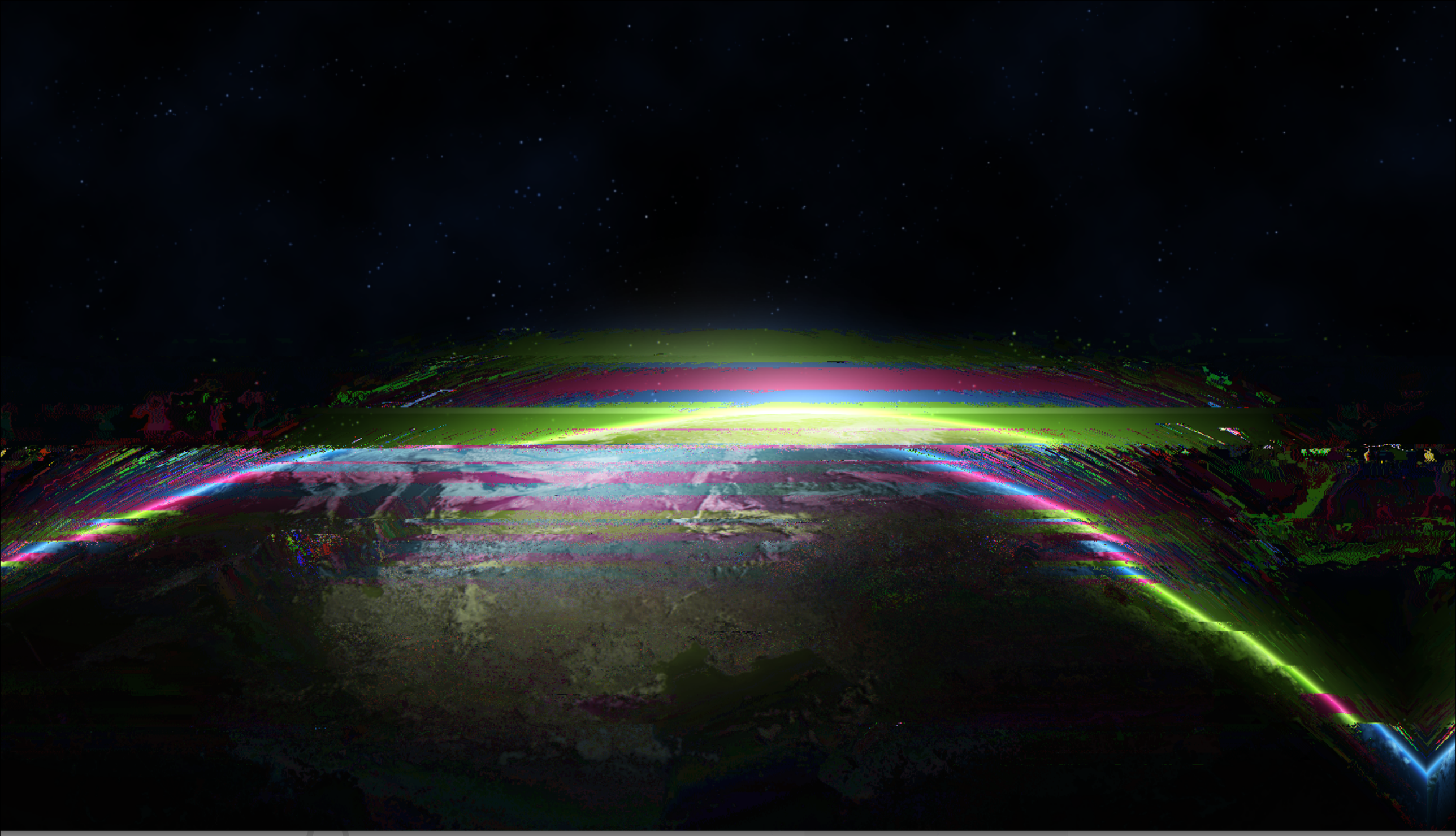
- V ee TA e e e he f a d a ed
  e d e ha 30 h a da he f
- N e e e i e f e ie , ea
  ha e e e i ge fe ac
- S e e e c i ici e e a e f i eed
  de e a hic e he



Current Work

Current Work

- B i d e f a a a i
- C e ec
  - P i g Pa b i ed Ja a a d Re de c i
  - ZO e a c g age e e Ma /Red ce c i e i
  - O i i a i e ea i g+Vi i a ica i



Questions?