

Or something similar as below:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vector3 Eye, At, Up;

Vector3 zaxis = At - Eye;

zaxis.Normalize();

Up.Normalize();

Vector3 xaxis = Vector3::Cross(Up, zaxis); xaxis.Normalize();

Vector3 yaxis = Vector3::Cross(zaxis, xaxis); yaxis.Normalize();

float r[16] =

{

xaxis.x, yaxis.x, -zaxis.x, 0,

xaxis.y, yaxis.y, -zaxis.y, 0,

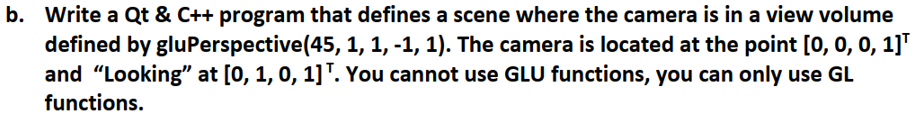
xaxis.z, yaxis.z, -zaxis.z, 0,

0, 0, 0, 1,

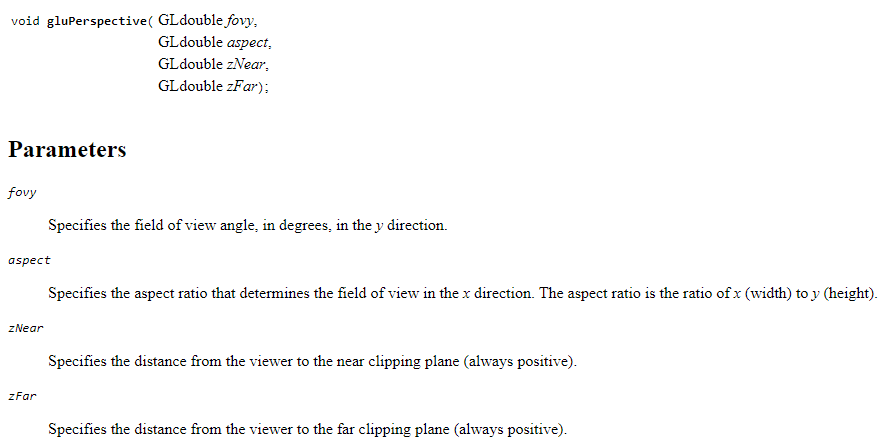
};

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*





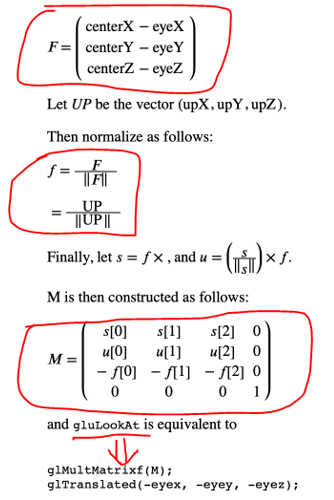
Definition:

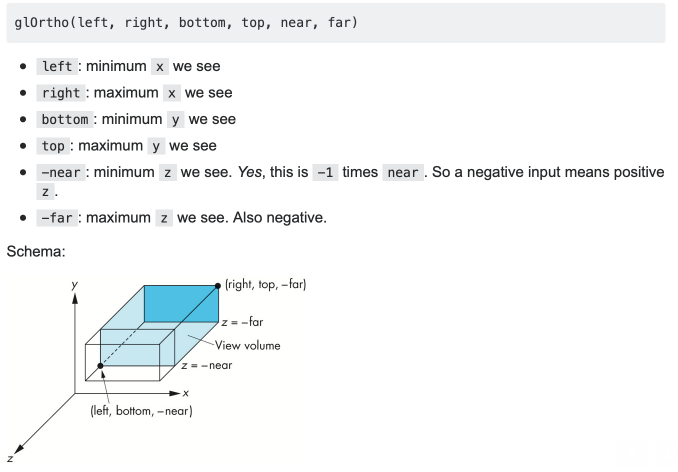


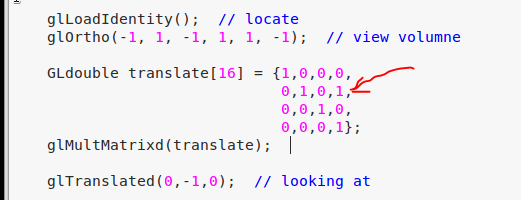
It means the view angle is 45, aspect ratio is 1, zNear is 1, zFar is -1

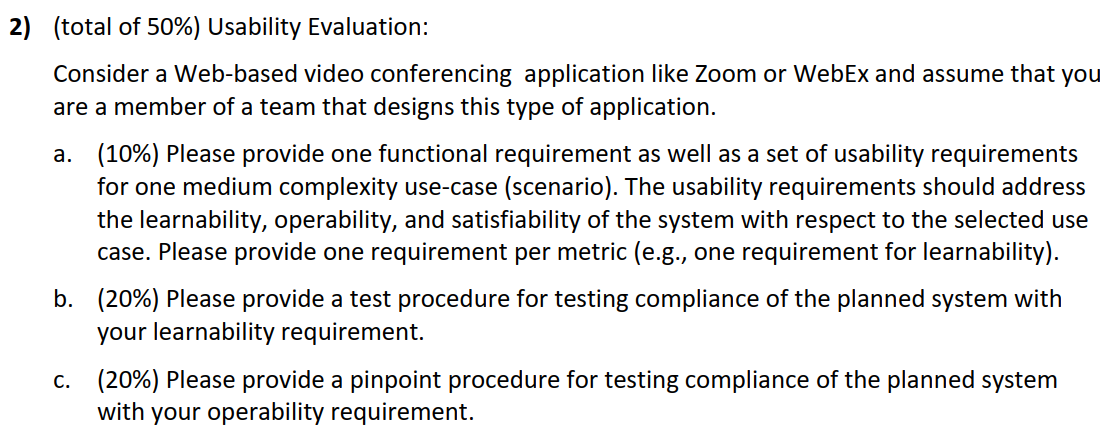
The camera is in origin: (0,0,0)

Looking at (0,1,0)











* Functional requirement:

1. A customer want to reserve the meeting room at CST from 4:00pm to 6:00pm with maximum attendee 300 people. The customer is a professor from Russian.
2. The need to configure the background color, image, sound, and check the availability to break the room into 50 small meeting room.
3. He need to login the website using mac computer and also windows computer at the same time.
4. He want to recording the whole meeting.
5. He does not read English. He only read Russian.

* Usability requirement

1. Effectiveness – Russian professors can complete reserving the meeting in less than 10 minutes, given a normal office environment. They will receive email notification within 10 second of reservation.
2. Efficiency – Given 2 expert attempting 10 tasks of setting up the meeting room according to specific requirement. The professor can finish them after he did repeatedly for 10 times (for 10 iid task), with the almost the same resource needed by the expert.
3. Satisfaction – The mean score (for future use of the software for 1 month) on the SUMI scale will be greater the 60.
4. Productivity - Given 2 expert attempting 10 tasks of setting up the meeting room according to specific requirement. Assume the time needed for each task is 3 minutes.

The professor can finishing the online book within 1 minutes, after he did repeatedly for 10 times (for 10 iid task)

1. Understandability – The professors will have less than 5% of errors of type 1 and less than 5% errors of type 2
2. Learnability - The average new user will reach the level of productive user after 10 number of

executions of each specific scenario based IID set of tasks.

1. Attractiveness - At least 80% of the users that have any experience with the system will rank the

system appeal level at 8 or above on a scale of 1 to 5

1. Now we need to time to test the learnability - The average new user will reach the level of productive user after 10 number of executions of each specific scenario based IID set of tasks.
2. Here we need to evaluation the software – online meeting software.
3. We identify a set of “scenarios”:

* Users from all of the world, speaking different language, want to reserve the meeting room at CST from 4:00pm to 6:00pm
* with minimum attendee 10 people, and maximum attendee 300 people.
* The user can configure the background color, image, sound, and check the availability to break the room into small meeting rooms.
* The meeting information can be sent to attendees.
* The zoom will set a password

1. Define 20 iid tasks (like above)

* Find a room with price lower than 500 USD while with 3 other roommates
* Find a room with price between 600-800 USD while with 1 other roommates
* Find an apartment with 3 rooms which near round rock
* …
* Etc.

1. We hire 20 people (new to the online meeting software) to take participant in the experiments
2. Those who complete all task within 30 minute will get two free movie tickets
3. The task procedure is to register the software, choose the meeting type, set the number of people, and send the invitation via email, URL, etc.
4. Record all the time needed (average time) to finish the tasks.
5. Try to find the curve and the power line
6. Also, we also hire 2 people (expert to the online booking meeting) to finish all the task. Find the base effort.
7. **Find the gap between base effort line and the power line. The gap is the learnability gap**
8. Produce data analysis report with explanation of the learnability difference between new users and expert

* Firstly we need the specific device to record the users’ activities during the experiment
* We need to analysis the fixation and saccades.
* For each video recording we segment the task with 10 seconds uniform time segments
* Pattern Recognitions for (classify interaction in time segments into several classes based on the level of effort) EE, NE (excessive effort) identify Excessive effort segments
* We ask the system expertise to find the metrics and do supervised classification
* We choose threshold-based method, and choose only one metrics: Number of saccades
* For example, one standard deviation above the average number of saccades (per segment) obtained in the training stage.
* Get the training subjects to perform the tasks, measure the number of saccades per segment per task per user and average over segments
* Label the segments where the result is above the threshold as E. Label the segments where the result is above the **threshold** as NE. (All segments are classified into Excessive or Non-excessive effort segment)
* Ask an expert to evaluate the segments classified as E.
* Get the segments that classified as E
* We focus on the E

The operability is like effectiveness: Given 2 expert attempting 10 tasks of setting up the meeting room according to specific requirement. The professor can finish them after he did repeatedly for 10 times (for 10 iid task), with the almost the same resource needed by the expert.

* Firstly, we define the two type of errors:
* Classifying NE segments as E segments is regarded as false positive or **type-I error**.
* Segments that show excessive effort per manual classification but are Identified excessive effort segment as non-excessive effort segments is regarded as false negative or **type-II error**.
* All segments are classified into Excessive or Non-excessive effort segment, which applied threshold “Number of saccades” during the test. On the other hand, we also consider below threshold:

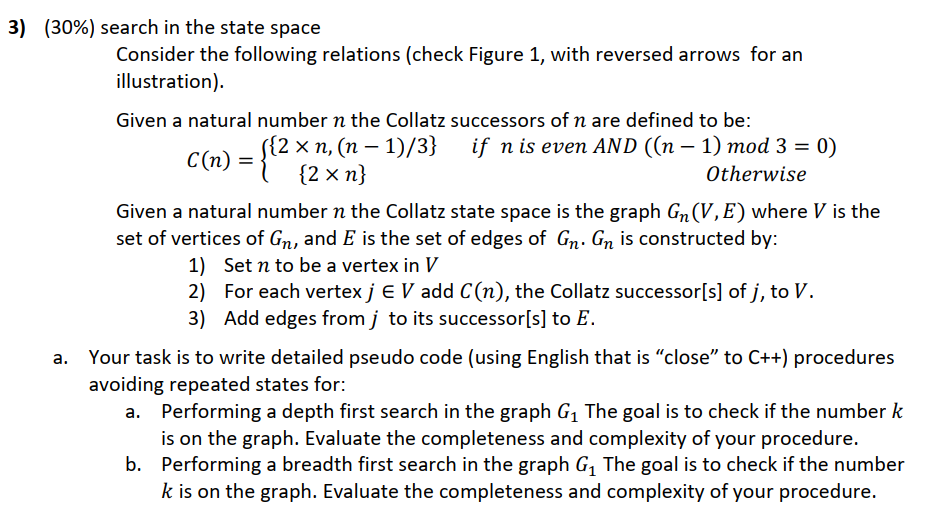
Average fixation duration

Number of fixation

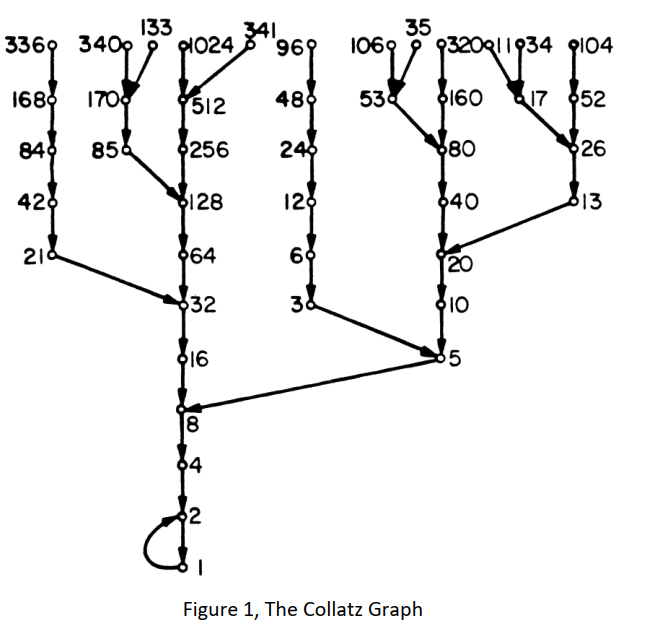
Average saccade amplitude

Average eye path traversed

* It is assumed that all the segments classified as Excessive Effort Segments are **due for an additional process of manual evaluation**.
* We then analysis the result based on the two type of error together with the threshold







Answer:

Firstly, we need to make a graph, given the number K, given the definition of:



Updated to: 

It means that,

* given 1, it will point to 2, 1
* given 2, it will point to 4, 1
* given 3, nothing happen
* given 4, nothing happen
* given 5, it will point to 10, 3

…

1. To solve this problem using BFS：
2. Define a set **SET**, which contains the nodes in the graph.
3. Define a visited set **SEEN**, those who have been visited, will be put into this set
4. Define a sub-class to validate if **k** already in the graph by:

Boolean **exist**(int k){

If ( (2\*n – 1)/3 == 0 && n%2==0) ) return true;

}

1. Corner case:

If(exist) return true;

1. Otherwise, we search in the range: [ K/2, 3\*K ]
3. Define a queue Q, to save the candidate node
5. Now begin the BFS
6. Push K/2 into the Q
7. while(true){

int cur = Q.pop();

SET.add(cur);

if cur == k, return true;

val = **exist(cur);**

if( ! val ) continue;

if(SEEN.add(2n) && 2n>k/2 && 2n<k\*3) Q.add(2n);

if(SEEN.add((2n-1)/3) && (2n-1)/3>k/2 && (2n-1)/3<k\*3 ) Q.add((2n-1)/3);

if(SEEN.add(( n+1 ) && (n+1)>k/2 && (n+1)<k\*3 ) Q.add((2n-1)/3);

}

return false;

* Completeness: if the value exists, my BFS code can definition find this solution
* Time complexity: O(n), n < 4\*K 🡪 every node will only be check for once

1. To solve this problem using DFS：

The variables are similar to BFS

Set<Integer> set // Define a set **SET**, which contains the nodes in the graph.

Set<Integer> seen // Define a visited set **SEEN**, those who have been visited, will be put into this set

Boolean **exist**(int k){

If ( (2\*n – 1)/3 == 0 && n%2==0) ) return true;

}

// Define a sub-class to validate if **k** already in the graph by:

If(exist(k)) return true; // Corner case:

Otherwise, we search in the range: [ K/2, 3\*K ]

Define a recursive function:

Boolean **findK**(int curr) {

If(!seen.add(cur)) continue;

If(**exist**(int curr)) return true;

If(cur <K/2 || cur >3\*K) return false;

Boolean a = **findK**( curr/2 );

Boolean b = **findK**((3\*curr + 1)/2);

If((2\*curr – 1)/3 == 0 && curr %2==0)){

Boolean a = **findK**( curr\*2 );

Boolean b = **findK**((2\*curr - 1)/3);

}

Return a || b;

}

findK(K);

* Completeness: if the value exists, my DFS code can definition find this solution
* Time complexity: O(n) 🡪 every node will only be check for once before program termination.