Γραφικά Υπολογιστών

Μάθημα 3 - Ένα υποτυπώδες σύστημα παρακολούθησης ακτίνων

Γιώργος Σφήκας

Basic Raytracing

- Θα δούμε τον αλγόριθμο «Raytracing» (παρακολούθηση ακτίνων) στην πιο υποτυπώδη, πρωτόλεια μορφή ...
- ...ωστόσο αυτή η μορφή θα είναι λειτουργική
- Υπενθύμιση: Μία από τις βασικές τεχνικές - λογικές rendering που θα δούμε στο μάθημα

Ζωγραφίζοντας ένα όμορφο τοπίο...

~~Ζωγραφίζοντας~~
Αποδίδοντας ένα όμορφο τοπίο

~~Ζωγραφίζοντας~~
Αποδίδοντας ένα όμορφο τοπίο

Αποδίδοντας ένα όμορφο τοπίο

- *Αλγόριθμος ζωγραφικής* για κάποιον που είναι εντελώς ατάλαντος ως ζωγράφος, πλην όμως εξαιρετικά μεθοδικός:

```
For each little square on the canvas
    Paint it the right color
```

- *ποιος πληροί αυτές τις προϋποθέσεις ;*

Αποδίδοντας ένα όμορφο τοπίο

- *Αλγόριθμος ζωγραφικής* για κάποιον που είναι εντελώς ατάλαντος ως ζωγράφος, πλην όμως εξαιρετικά μεθοδικός:

```
For each little square on the canvas
    Paint it the right color
```

- *ποιος πληροί αυτές τις προϋποθέσεις ;*

Αποδίδοντας ένα όμορφο τοπίο

- Το μόνο πρόβλημα είναι ότι πρέπει να γίνουμε όσο πιο λεπτομερείς γίνεται -- ο Η/Υ δεν θα μας καταλάβει αλλιώς.

```
For each little square on the canvas
    Paint it the right color
```

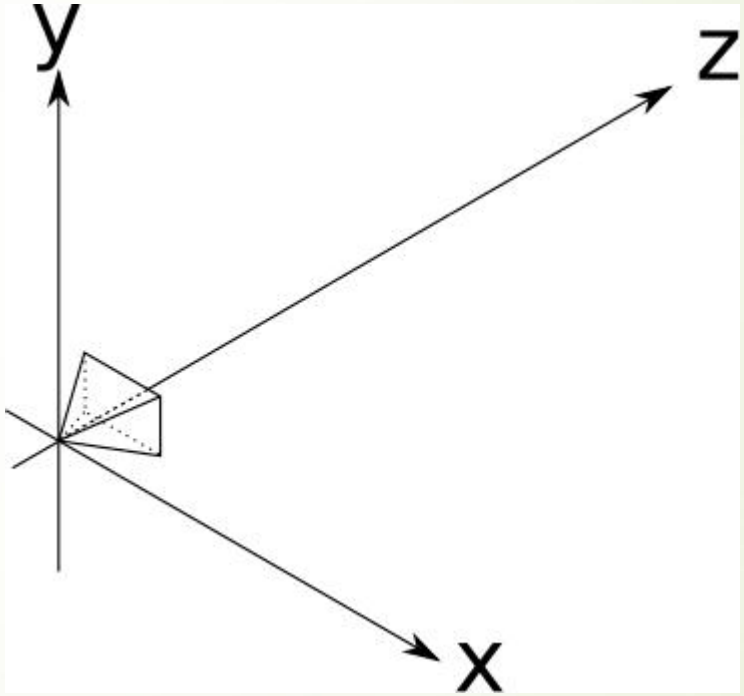
Αποδίδοντας ένα όμορφο τοπίο

- Το μόνο πρόβλημα είναι ότι πρέπει να γίνουμε όσο πιο λεπτομερείς γίνεται -- ο Η/Υ δεν θα μας καταλάβει αλλιώς.

```
Place the eye and the frame as desired
For each square on the canvas
    Determine which square on the grid corresponds to this square on the canvas
    Determine the color seen through that grid square
    Paint the square with that color
```

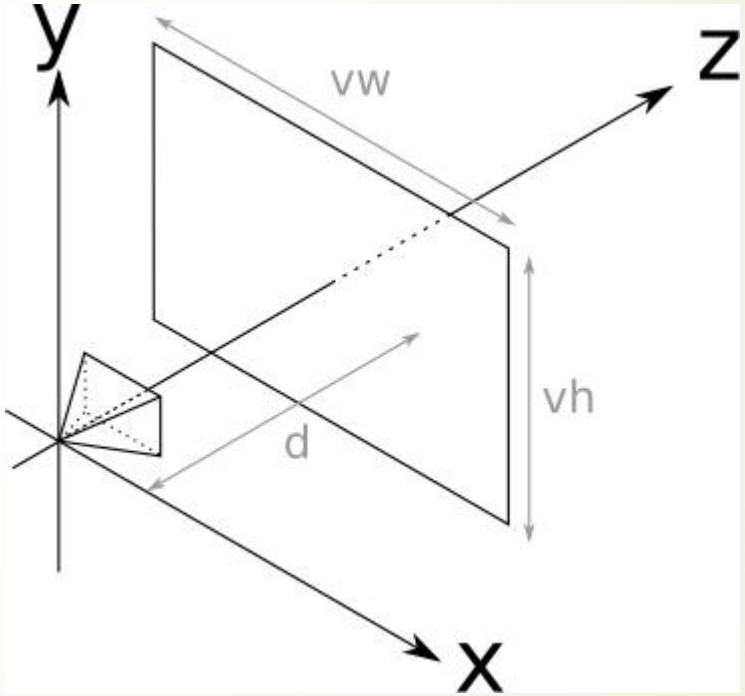
Πριν προχωρήσουμε, κάποιες απλουστευτικές υποθέσεις

- Σταθερή θέση θέασης στον χώρο. Είναι η θέση της “κάμερας - οφθαλμού” (camera position)
- Θα θεωρήσουμε ότι ταυτίζεται με την αρχή των αξόνων
- Σταθερή γωνία θέασης
- (Συγγενής έννοια στην φωτογραμμετρία: εξωτερικός προσανατολισμός)



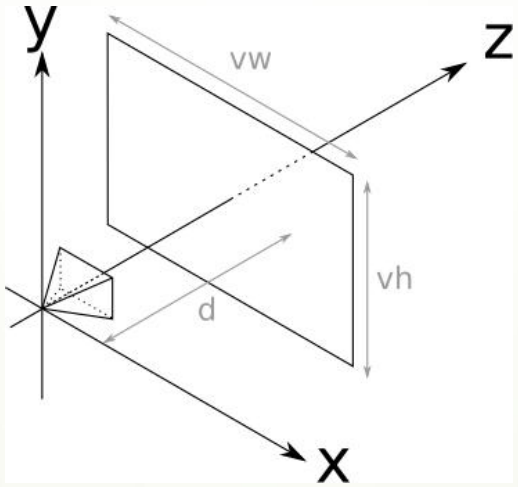
Πριν προχωρήσουμε, κάποιες απλουστευτικές υποθέσεις

- «Viewport» (πεδίο παράστασης)
- (Συγγενής έννοια στην Φωτογραμμετρία: Το επίπεδο προβολής)
- Το μέγεθος του viewport και η απόστασή του από την κάμερα ορίζουν το «field of view (FOV)» (πεδίο θέασης, μετράται σε μοίρες)



Πριν προχωρήσουμε, κάποιες απλουστευτικές υποθέσεις

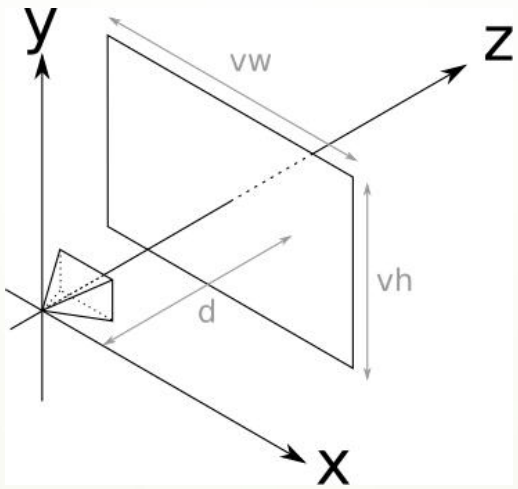
```
❶Place the camera and the viewport as desired
For each pixel on the canvas
    ❷Determine which square on the viewport corresponds to this pixel
    ❸Determine the color seen through that square
    ❹Paint the pixel with that color
```



Πριν προχωρήσουμε, κάποιες απλουστευτικές υποθέσεις

```
❶Place the camera and the viewport as desired
For each pixel on the canvas
    ❷Determine which square on the viewport corresponds to this pixel
    ❸Determine the color seen through that square
    ❹Paint the pixel with that color
```

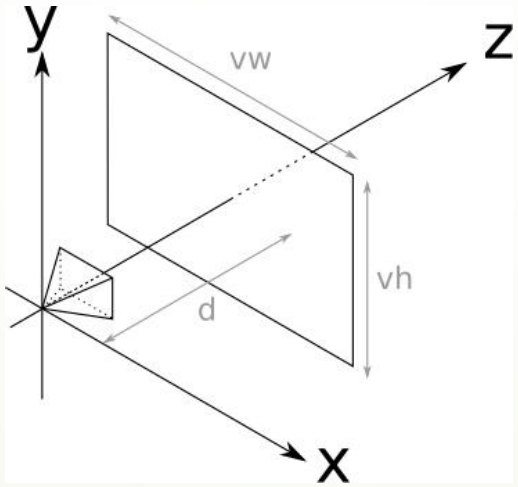
Κι όμως έχουμε ήδη τελειώσει με δύο από αυτά τα βήματα!



Πριν προχωρήσουμε, κάποιες απλουστευτικές υποθέσεις

```
❶Place the camera and the viewport as desired
For each pixel on the canvas
    ❷Determine which square on the viewport corresponds to this pixel
    ❸Determine the color seen through that square
    ❹Paint the pixel with that color
```

Κι όμως έχουμε ήδη τελειώσει με δύο από αυτά τα βήματα!



- (1) το είδαμε προηγουμένως
- (4) putPixel
- Μένει το (2) και το (3)!

Αντιστοιχία pixels (Cx,Cy) με θέσεις στο viewport (Vx, Vy, Vz)

```
❶Place the camera and the viewport as desired
For each pixel on the canvas
    ❷Determine which square on the viewport corresponds to this pixel
    ❸Determine the color seen through that square
    ❹Paint the pixel with that color
```

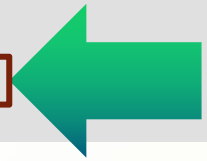
$$V_x = C_x \frac{V_w}{C_w}$$

$$V_y = C_y \frac{V_h}{C_h}$$

$$V_z = d$$

Προσδιορισμός χρώματος με παρακολούθηση ακτίνας

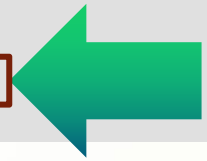
```
❶Place the camera and the viewport as desired
For each pixel on the canvas
    ❷Determine which square on the viewport corresponds to this pixel
    ❸Determine the color seen through that square
    ❹Paint the pixel with that color
```



- Μια πιστή προσομοίωση της φυσικής διαδικασίας θα προϋπέθετε να παρακολουθήσουμε τις φωτεινές ακτίνες από την πηγή τους, μέχρι τον οφθαλμό - κάμερα...

Προσδιορισμός χρώματος με παρακολούθηση ακτίνας

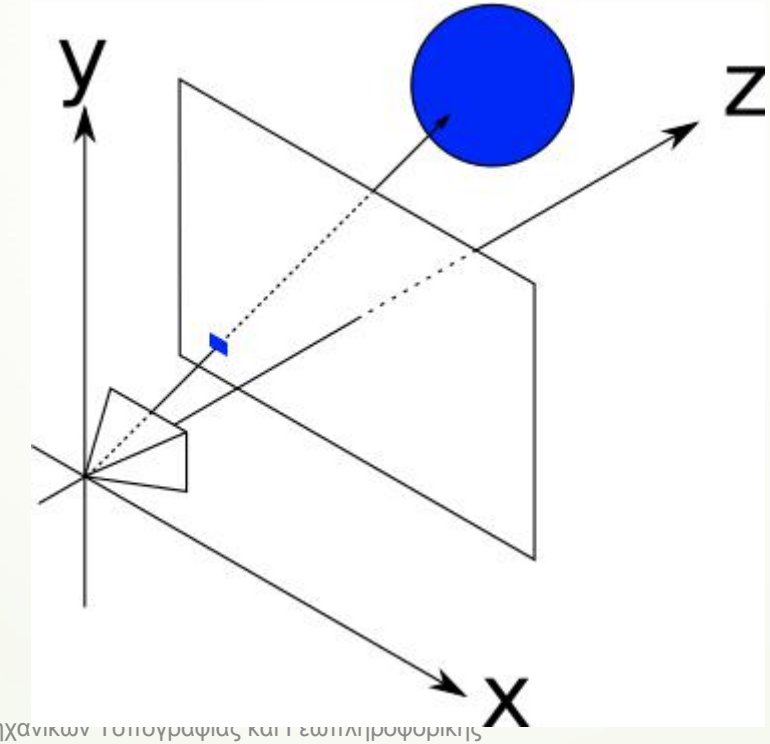
```
❶Place the camera and the viewport as desired
For each pixel on the canvas
    ❷Determine which square on the viewport corresponds to this pixel
    ❸Determine the color seen through that square
    ❹Paint the pixel with that color
```



- Μια πιστή προσομοίωση της φυσικής διαδικασίας θα προϋπέθετε να παρακολουθήσουμε τις φωτεινές ακτίνες από την πηγή τους, μέχρι τον οφθαλμό - κάμερα...
- Στο raytracing “κλέβουμε”, κάνοντας ακριβώς το ανάποδο: παρακολουθούμε ακτίνες αρχίζοντας *από* τον οφθαλμό - κάμερα προς την πηγή

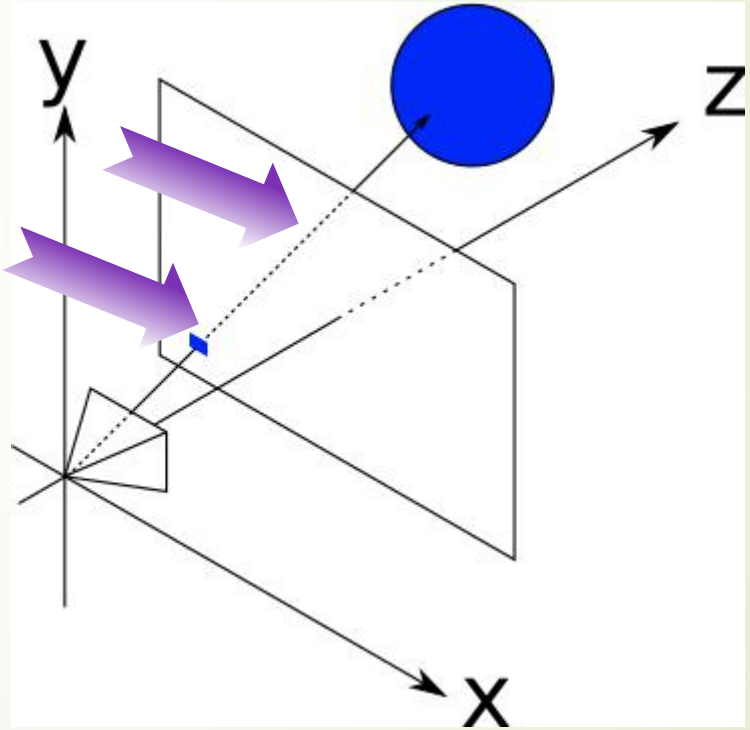
Προσδιορισμός χρώματος με παρακολούθηση ακτίνας

```
❶Place the camera and the viewport as desired
For each pixel on the canvas
    ❷Determine which square on the viewport corresponds to this pixel
    ❸Determine the color seen through that square
    ❹Paint the pixel with that color
```



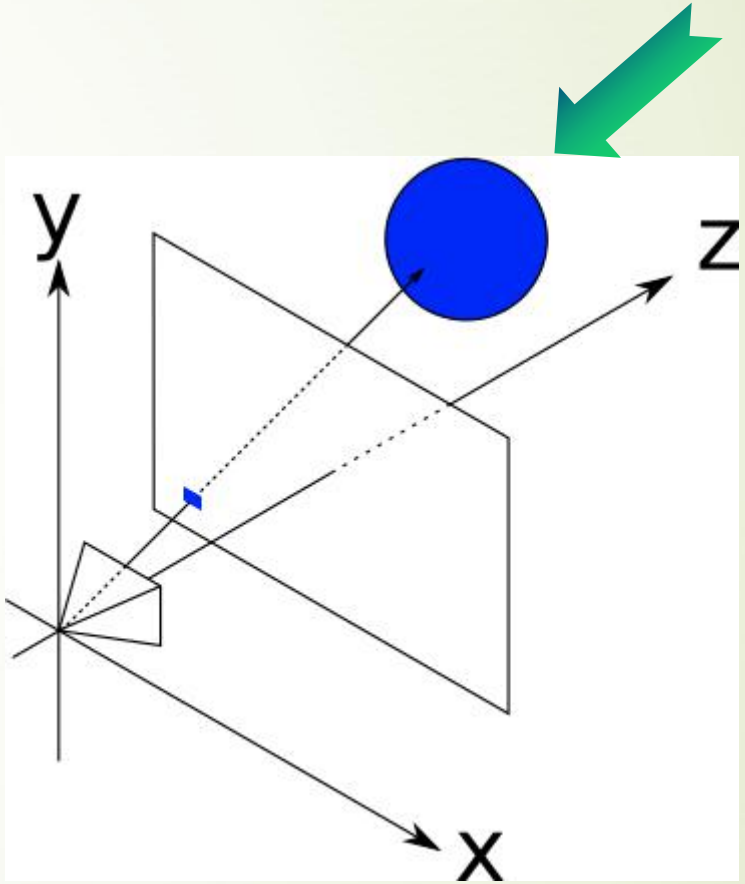
Προσδιορισμός χρώματος με παρακολούθηση ακτίνας

- Για την “παρακολούθηση” χρειαζόμαστε:
  - Έναν τρόπο να περιγράψουμε την ακτίνα



Προσδιορισμός χρώματος με παρακολούθηση ακτίνας

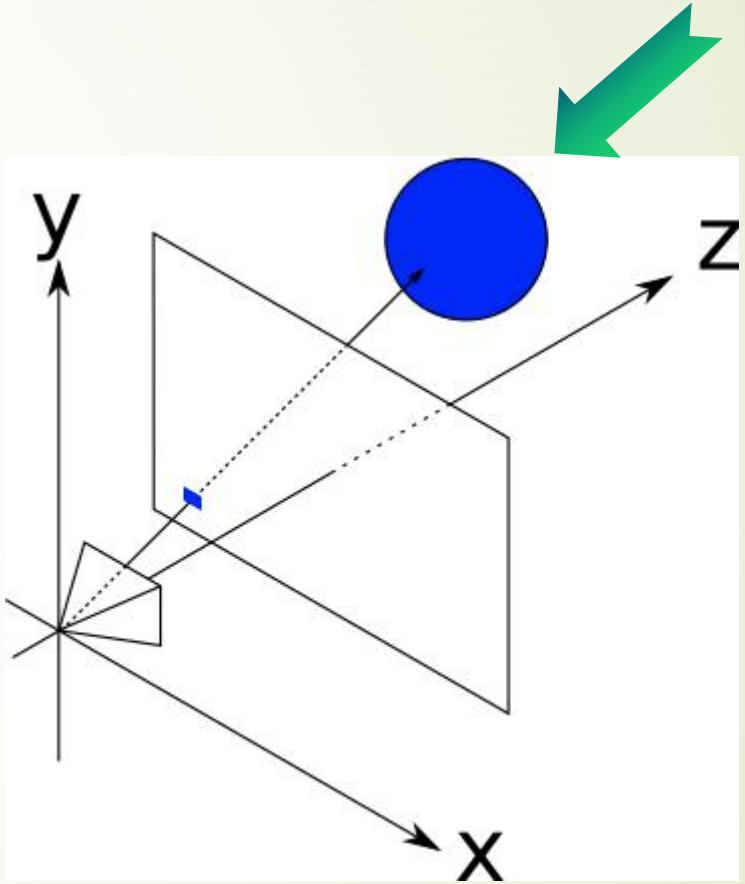
- Για την “παρακολούθηση” χρειαζόμαστε:
  - Έναν τρόπο να περιγράψουμε την ακτίνα
  - Έναν τρόπο να περιγράψουμε κάθε αντικείμενο της σκηνής
  - Έναν τρόπο να ελέγξουμε πότε η ακτίνα τέμνει κάποιο αντικείμενο



Προσδιορισμός χρώματος με παρακολούθηση ακτίνας

Εξίσωση ευθείας στον χώρο

- Για την “παρακολούθηση” χρειαζόμαστε:
  - Έναν τρόπο να περιγράψουμε την ακτίνα
  - Έναν τρόπο να περιγράψουμε κάθε αντικείμενο της σκηνής
  - Έναν τρόπο να ελέγξουμε πότε η ακτίνα τέμνει κάποιο αντικείμενο

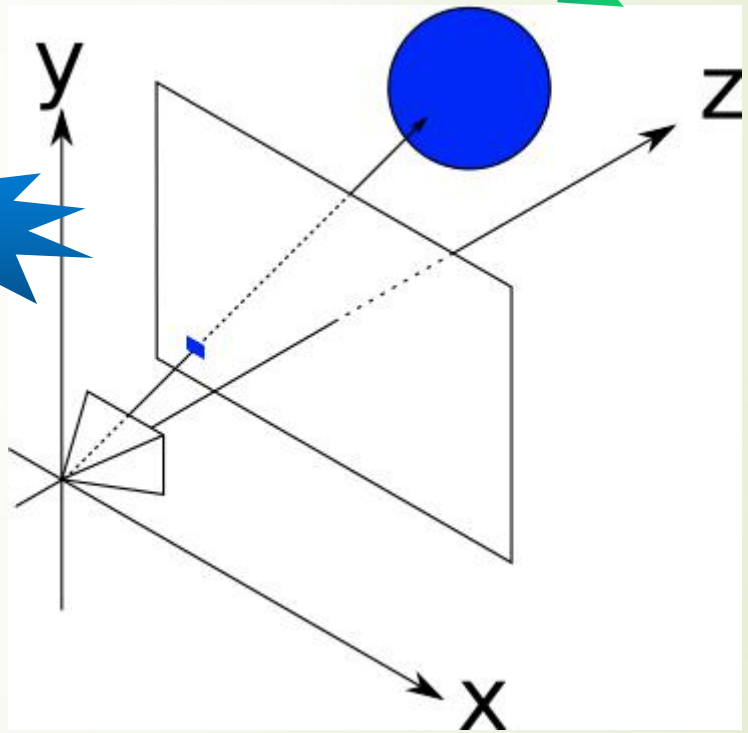


Προσδιορισμός χρώματος με παρακολούθηση ακτίνας

- Για την “παρακολούθηση” χρειαζόμαστε:
  - Έναν τρόπο να περιγράψουμε την ακτίνα
  - Έναν τρόπο να περιγράψουμε κάθε αντικείμενο της σκηνής
  - Έναν τρόπο να ελέγξουμε πότε η ακτίνα τέμνει κάποιο αντικείμενο

Εξίσωση ευθείας στον χώρο

Εξίσωση σφαίρας



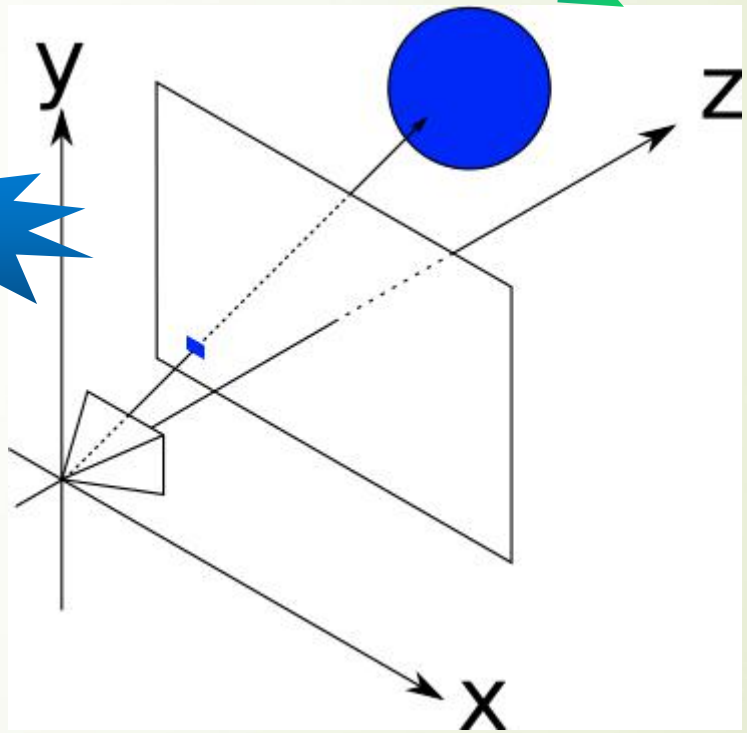
Προσδιορισμός χρώματος με παρακολούθηση ακτίνας

- Για την “παρακολούθηση” χρειαζόμαστε:
  - Έναν τρόπο να περιγράψουμε την ακτίνα
  - Έναν τρόπο να περιγράψουμε κάθε αντικείμενο της σκηνής
  - Έναν τρόπο να ελέγξουμε πότε η ακτίνα τέμνει κάποιο αντικείμενο

Εξίσωση ευθείας στον χώρο

Εξίσωση σφαίρας

Λύση συστήματος εξισώσεων



Ray equation (Εξίσωση ακτίνας)

- Είναι απλά μια εξίσωση ευθείας στον χώρο
- Γενικά, για ευθεία που περνά από Α,Β:

$$\overrightarrow{OP} = \overrightarrow{OA} + t\overrightarrow{AB}$$

Ray equation (Εξίσωση ακτίνας)

- Είναι απλά μια εξίσωση ευθείας στον χώρο
- Γενικά, για ευθεία που περνά από Α,Β:

$$\overrightarrow{OP} = \overrightarrow{OA} + t\overrightarrow{AB}$$

- Αν το ένα σημείο είναι η αρχή των αξόνων, μπορώ να γράψω:

$$\overrightarrow{OP} = t\overrightarrow{\alpha}$$

Ray equation (Εξίσωση ακτίνας)

- Είναι απλά μια εξίσωση ευθείας στον χώρο
- Γενικά, για ευθεία που περνά από Α,Β:

$$\overrightarrow{OP} = \overrightarrow{OA} + t\overrightarrow{AB}$$

- Αν το ένα σημείο είναι η αρχή των αξόνων, μπορώ να γράψω:

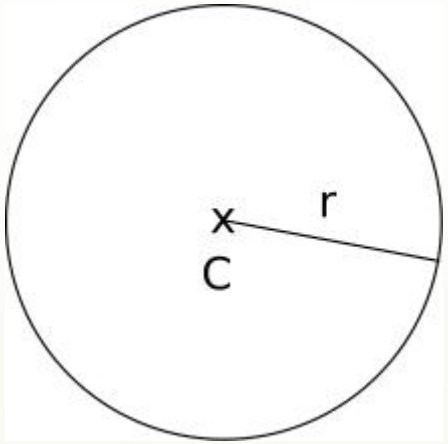
$$\overrightarrow{OP} = t\overrightarrow{\alpha}$$

- ...όπου $\overrightarrow{\alpha}$ έχει αρχικό σημείο Ο και ως τελικό κάποιο σημείο του viewport.

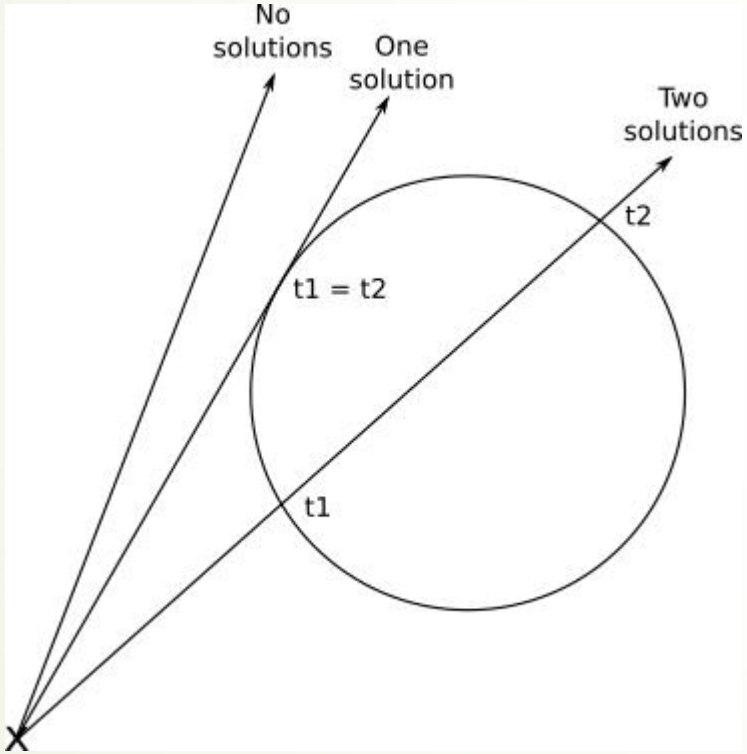
Sphere equation (Εξίσωση σφαίρας)

- Σφαίρα = γεωμετρικός τόπος σημείων που απέχουν σταθερή απόσταση r από σημείο C

$$|\overrightarrow{OP} - \overrightarrow{OC}| = r$$

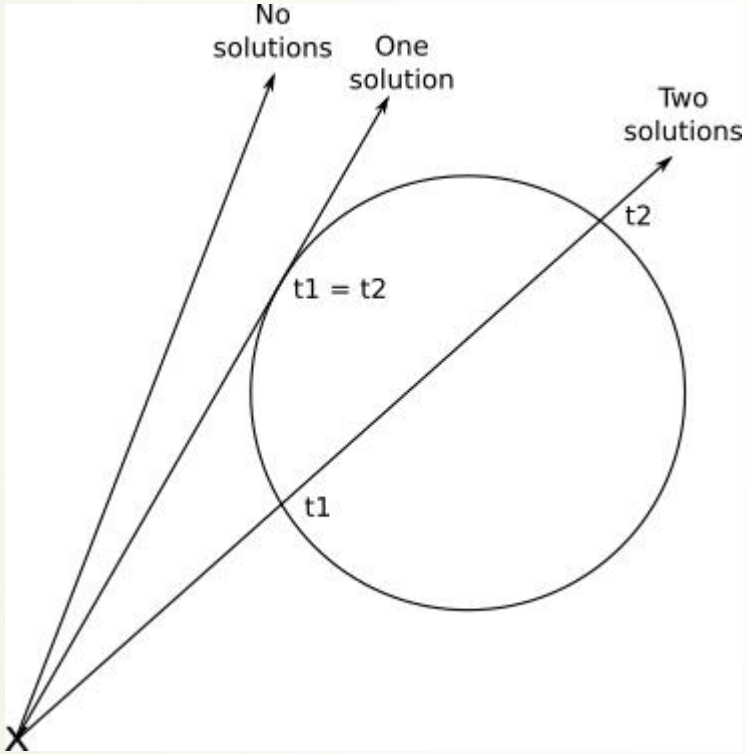


Ray meets Sphere



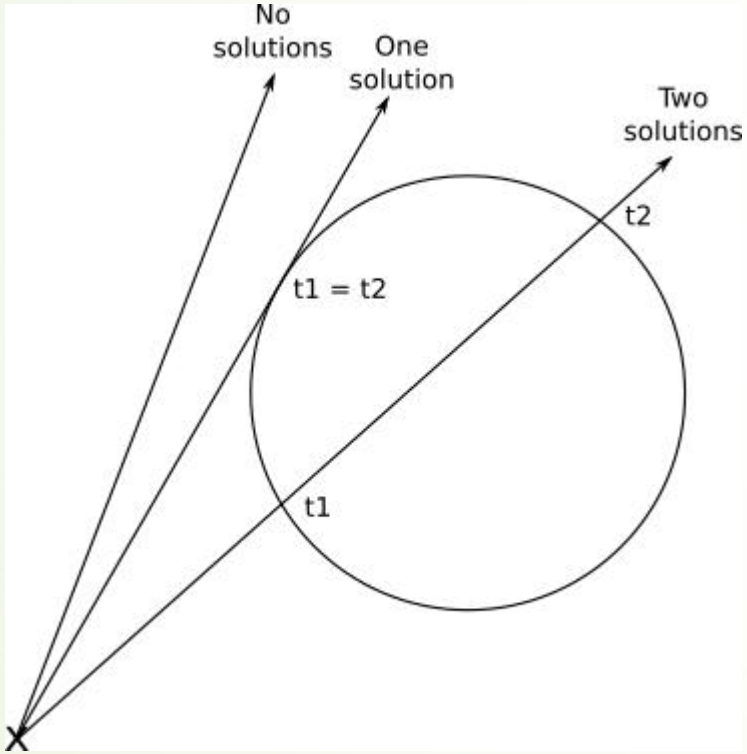
- Καταλήγουμε σε μια δευτεροβάθμια εξίσωση ...
- Μία εξίσωση για *κάθε ακτίνα και κάθε σφαίρα*
- Μια (πραγματική) λύση *t* μας δίνει το σημείο τομής, $t\vec{\alpha}$
- Θέλουμε μόνο πραγματικές λύσεις;

Ray meets Sphere



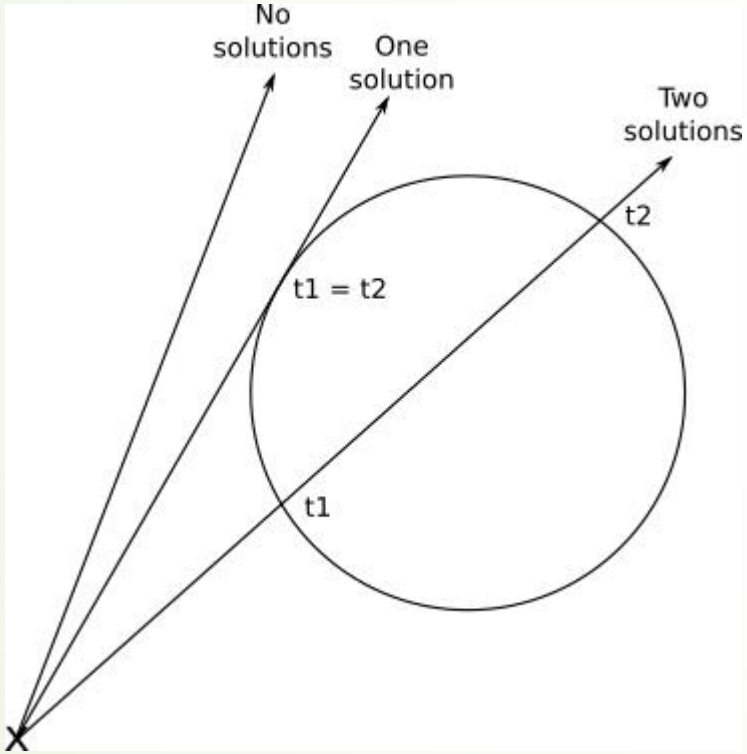
- Καταλήγουμε σε μια δευτεροβάθμια εξίσωση ...
- Μία εξίσωση για *κάθε ακτίνα και κάθε σφαίρα*
- Μια (πραγματική) λύση *t* μας δίνει το σημείο τομής, $t\vec{\alpha}$
- Θέλουμε μόνο πραγματικές λύσεις; (ναι)

Ray meets Sphere



- Καταλήγουμε σε μια δευτεροβάθμια εξίσωση ...
- Μία εξίσωση για *κάθε ακτίνα και κάθε σφαίρα*
- Μια (πραγματική) λύση *t* μας δίνει το σημείο τομής, $t\vec{\alpha}$
- Θέλουμε μόνο πραγματικές λύσεις; (ναι)
- Θέλουμε μόνο λύσεις σε ένα συγκεκριμένο εύρος για το t ?

Ray meets Sphere



- Καταλήγουμε σε μια δευτεροβάθμια εξίσωση ...
- Μία εξίσωση για *κάθε ακτίνα και κάθε σφαίρα*
- Μια (πραγματική) λύση *t* μας δίνει το σημείο τομής, $t\vec{\alpha}$
- Θέλουμε μόνο πραγματικές λύσεις; (ναι)
- Θέλουμε μόνο λύσεις σε ένα συγκεκριμένο εύρος για το t ? (ναι)

Η τελική μορφή του αλγόριθμου

```
❶Place the camera and the viewport as desired
For each pixel on the canvas
    ❷Determine which square on the viewport corresponds to this pixel
    ❸Determine the color seen through that square
    ❹Paint the pixel with that color
```

Η τελική μορφή του αλγόριθμου

```
❶Place the camera and the viewport as desired
For each pixel on the canvas
    ❷Determine which square on the viewport corresponds to this pixel
    ❸Determine the color seen through that square
    ❹Paint the pixel with that color
```

```
O = (0, 0, 0)
for x = -Cw/2 to Cw/2 {
    for y = -Ch/2 to Ch/2 {
        D = CanvasToViewport(x, y)
        color = TraceRay(O, D, 1, inf)
        canvas.PutPixel(x, y, color)
    }
}
```

Η τελική μορφή του αλγόριθμου

```
❶Place the camera and the viewport as desired
For each pixel on the canvas
    ❷Determine which square on the viewport corresponds to this pixel
    ❸Determine the color seen through that square
    ❹Paint the pixel with that color
```

```
O = (0, 0, 0)
for x = -Cw/2 to Cw/2 {
    for y = -Ch/2 to Ch/2 {
        D = CanvasToViewport(x, y)
        color = TraceRay(O, D, 1, inf)
        canvas.PutPixel(x, y, color)
    }
}
```

- όπου…

```
CanvasToViewport(x, y) {
    return (x*Vw/Cw, y*Vh/Ch, d)
}
```

- και μας λείπει η συνάρτηση TraceRay.

Η τελική μορφή του αλγόριθμου

```
O = (0, 0, 0)
for x = -Cw/2 to Cw/2 {
    for y = -Ch/2 to Ch/2 {
        D = CanvasToViewport(x, y)
        color = TraceRay(O, D, 1, inf)
        canvas.PutPixel(x, y, color)
    }
}
```

```
TraceRay(O, D, t_min, t_max) {
    closest_t = inf
    closest_sphere = NULL
    for sphere in scene.spheres {
        t1, t2 = IntersectRaySphere(O, D, sphere)
        if t1 in [t_min, t_max] and t1 < closest_t {
            closest_t = t1
            closest_sphere = sphere
        }
        if t2 in [t_min, t_max] and t2 < closest_t {
            closest_t = t2
            closest_sphere = sphere
        }
    }
    if closest_sphere == NULL {
       ❶return BACKGROUND_COLOR
    }
    return closest_sphere.color
}
```

Η τελική μορφή του αλγόριθμου

```
O = (0, 0, 0)
for x = -Cw/2 to Cw/2 {
    for y = -Ch/2 to Ch/2 {
        D = CanvasToViewport(x, y)
        color = TraceRay(O, D, 1, inf)
        canvas.PutPixel(x, y, color)
    }
}
```

```
TraceRay(O, D, t_min, t_max) {
    closest_t = inf
    closest_sphere = NULL
    for sphere in scene.spheres {
        t1, t2 = IntersectRaySphere(O, D, sphere)
        if t1 in [t_min, t_max] and t1 < closest_t {
            closest_t = t1
            closest_sphere = sphere
        }
        if t2 in [t_min, t_max] and t2 < closest_t {
            closest_t = t2
            closest_sphere = sphere
        }
    }
    if closest_sphere == NULL {
        return BACKGROUND_COLOR
    }
    return closest_sphere.color
}
```

Η τελική μορφή του αλγόριθμου

- Η IntersectRaySphere απλά υπολογίζει τομές λύνοντας μια δευτεροβάθμια εξίσωση.

```
IntersectRaySphere(O, D, sphere) {
    r = sphere.radius
    CO = O - sphere.center

    a = dot(D, D)
    b = 2*dot(CO, D)
    c = dot(CO, CO) - r*r

    discriminant = b*b - 4*a*c
    if discriminant < 0 {
        return inf, inf
    }

    t1 = (-b + sqrt(discriminant)) / (2*a)
    t2 = (-b - sqrt(discriminant)) / (2*a)
    return t1, t2
}
```

Περιγραφή της σκηνής

- Το μόνο που μένει είναι να βάλουμε μερικά αντικείμενα στον χώρο!
- Για παράδειγμα ορίζουμε 3 σφαίρες:

```
viewport_size = 1 x 1
projection_plane_d = 1
sphere {
    center = (0, -1, 3)
    radius = 1
    color = (255, 0, 0)  # Red
}
sphere {
    center = (2, 0, 4)
    radius = 1
    color = (0, 0, 255)  # Blue
}
sphere {
    center = (-2, 0, 4)
    radius = 1
    color = (0, 255, 0)  # Green
}
```

- Εκτελούμε τον κώδικά μας, και.. η συνέχεια επί της οθόνης!

Ανακεφαλαίωση

- Περιγράψαμε έναν αλγόριθμο παρακολούθησης ακτίνων στην πιο υποτυπώδη του μορφή
- Όλα τα βασικά συστατικά είναι εδώ : Πρώτα και κύρια, η λογική του raytracing. Παρακολουθούμε τις ακτίνες “κλέβοντας”, δηλαδή αρχίζουμε από το μάτι και πάμε προς την φωτεινή πηγή

Στο αυριανό εργαστήριο..

- Θα υλοποιήσουμε ό,τι περιγράψαμε!

Μελέτη στο βιβλίο - πηγές

- G.Gambetta, Basic Raytracing (https://gabrielgambetta.com/computer-graphics-from-scratch/02-basic-raytracing.html)

Ερωτήσεις ... ;;;