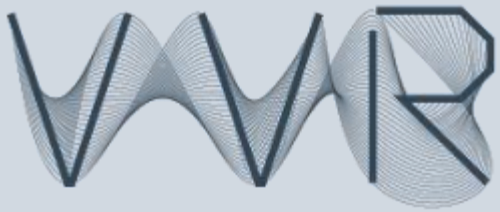

# VVR Framework

DEVELOPER'S GUIDE …

# class vvr::Scene

- **Αναλαμβάνει την διαχείριση της σκηνής (2D/3D).**
  - ✓ Δημιουργία παραθύρου
  - ✓ Render
  - ✓ Event handling
- **Για να δημιουργήσουμε την σκηνή μας:**
  - ✓ Ορίζουμε C++ κλάση που κληρονομεί την κλάση Scene.
- **Για να απεικονίσουμε γεωμετρικά αντικείμενα:**
  - ✓ Υλοποιούμε την μέθοδο void draw();
- **Λειτουργίες που πρέπει να εκτελεστούν μία φορά στην αρχή:**
  - ✓ Τοποθετούνται στον Constructor της κλάσης. (Αρχικοποιήσεις μεταβλητών,…).
- **Για να αποκριθούμε σε User Events (*mouse, keyboard*):**
  - ✓ Υλοποιούμε τις αντίστοιχες μεθόδους.

```
class Scene01 : public vvr::Scene
{
  public:
  Scene01();
  void draw() override;
  ...
}
```

# int mainLoop (int argc, char* argv[], Scene *scene)

- **Αποτελεί το σημείο έναρξης της εφαρμογής.**
  - ✓ Από την κλήση της και μέχρι τον τερματισμό του προγράμματος ο έλεγχος της ροής *παραχωρείται* στην βιβλιοθήκη.
  - ✓ Δικός μας κώδικας εκτελείται μόνο μέσω:
    - ✓ Των μεθόδων που υλοποιούμε στο αντικείμενο Scene* που περνάμε ως παράμετρο στην mainLoop().

# Drawing Σχημάτων

- **Κλάσεις που διευκολύνουν την απεικόνιση βασικών σχημάτων:**
  - ✓ `struct Point2D`
  - ✓ `struct LineSeg2D`
  - ✓ `struct Line2D`
  - ✓ `struct Circle2D`
  - ✓ `struct Triangle2D`
- Όλες είναι απόγονοι του struct Shape
  - ✓ Στον Constructor καθορίζονται {Χρώμα, Θέση}
    Π.χ.: *Circle2D circle ( x, y, rad,* ***Colour::yellow****);*
  - ✓ Γίνονται render με κλήση της draw() - **Μέσα από την Scene::draw() !!!**

```
void Scene01::draw()
{
  Circle2D circle = Circle2D( x,y,r, Colour::yellow);
  circle.draw();
  ...
}
```

# Event Handling

- Για να αποκριθούμε σε ενέργειες του χρήστη (mouse / keyboard) υλοποιούμε τις παρακάτω μεθόδους:
  - ✓ `void mousePressed(int x, int y, int modif) override;`
  - ✓ `void mouseMoved(int x, int y, int modif) override;`
  - ✓ `void mouseWheel(int dir, int modif) override;`
- Οι παραπάνω συναρτήσεις καλούνται αυτόματα.
- Τα ***x,y*** περιέχουν τις συντεταγμένες του mouse **σε pixel** την στιγμή του event.
- Το origin *(0,0)* βρίσκεται στο κέντρο του παραθύρου.
- Το **modif** είναι bitwise flag που έχει άσσους σε προκαθορισμένα bits αν κάποιο από τα πλήκτρα (ctrl, shift, alt, …) ήταν πατημένο την στιγμή του event.
- Το **dir** συμβολίζει την κατεύθυνση της ρόδας του mouse. (+/-).

# Event Handling (Default implementation)

- **Scene::reset**
  - Καλείται με πάτημα του πλήκτρου '**R**'.
  - Η κλήση γίνεται μέσα από την **Scene::keyEvent**
  - Η default reset το μόνο που κάνει είναι να επαναφέρει τον αρχικό προσανατολισμό. (*3D μόνο*).
- **Scene::mousePressed / Scene:: mouseMoved**
  - Προσανατολισμός κάμερας. (*3D μόνο*).
- **ΣΗΜΑΝΤΙΚΟ**:
- *Για να μην χάσουμε την default συμπεριφορά όταν κάνουμε override, πρέπει να καλέσουμε την ίδια μέθοδο της base class.*
- *Π.χ. Αν θέλουμε να «πιάσουμε» τα events mouse click* αλλά να μην χάσουμε την λειτουργικότητα στριψίματος κάμερας κάνουμε το εξής:

```
void Scene02::mousePressed(int x, int y, int modif)
{
  Scene::mousePressed(x, y, modif);
  …
  …
}
```