Κεφάλαιο 2ο
Είσοδος/Εξοδος – Διαχείριση γεγονότων - Αποκοπή στις δύο διαστάσεις

Εισαγωγή

Στο Κεφάλαιο αυτό παρουσιάζουμε τις εντολές μέσω των οποίων εκτελούνται στην OpenGL διαδικασίες εισόδου/εξόδου. Σε ότι αφορά τις διαδικασίες εξόδου, αναλύονται οι ρουτίνες που χρησιμοποιούνται για τη δημιουργία παραθύρων, αλλά και για την επιλογή πιο σύνθετων ρυθμίσεων (όπως λ.χ. για την εφαρμογή διαφορετικών χρωματικών μοντέλων ή για τη δημιουργία κινουμένων γραφικών). Αναλύεται η αρχή λειτουργίας των προγραμμάτων στην OpenGL ως μια συνεχής διαδικασία ακρόασης και ανταπόκρισης σε γεγονότα. Με τον όρο γεγονότα, αναφερόμαστε σε ειδοποιήσεις που εγείρονται, είτε από το λειτουργικό σύστημα, είτε από το χρήστη μέσω συσκευών εισόδου. Συνεπώς, η διαχείριση γεγονότων παρέχει ένα μηχανισμό μέσω του οποίου μπορούν να σχεδιαστούν αλληλεπιδραστικές εφαρμογές. Επιπλέον επεξηγούνται οι διαδικασίες που εμπλέκονται στη διαδικασία αναπαράστασης μιας σκηνής σε συσκευές εξόδου και αναλύονται οι αλγόριθμοι αποκοπής και μετασχηματισμού παρατήρησης (οι οποίοι αντιστοιχούν τις θέσεις των σημείων της σκηνής στις αντίστοιχες θέσεις τους στη συσκευή εξόδου).

2.1 Σχηματισμός παραθύρων

Η δημιουργία παραθύρων, η απεικόνιση γραφικών σκηνών και γενικότερα οι διεργασίες εισόδου-εξόδου, είναι διαδικασίες που εξαρτώνται από την αρχιτεκτονική του συστήματος (platform-dependent). Όπως αναφέρθηκε στην Εισαγωγή, οι εντολές που εκτελούν τις συγκεκριμένες διαδικασίες περιλαμβάνονται στη βιβλιοθήκη GLUT.

Για τη χρήση εντολών της βιβλιοθήκης GLUT αρχικά πρέπει να δώσουμε την εντολή αρχικοποίησής της ***glutInit***:

void glutInit(int argc, char **argv);

όπου *argc* και *argv* το πλήθος και οι τιμές των ορισμάτων που περνάμε στο πρόγραμμα μέσω της γραμμής εντολών (command line arguments). Το πρώτο όρισμα (argv[0]) επιστρέφει το όνομα του εκτελέσιμου αρχείου.

Π.χ. εάν το εκτελέσιμο αρχείο έχει το όνομα MyOpenGLApp και δώσουμε στη γραμμή εντολών DOS:

```
MyOpenGLApp someArgument1 someArgument2
```

Τότε το όρισμα argv[1] θα έχει την τιμή “someArgument1” και το όρισμα argv[2] θα έχει την τιμή “someArgument2”.

Η θέση στην οθόνη, στην οποία θα εμφανιστεί το παράθυρο της εφαρμογής μας , καθορίζεται με την εντολή ***glutInitWindowPosition***.

void glutInitWindowPosition(int x, int y);

όπου *x,y* οι συντεταγμένες του άνω αριστερού άκρου του παραθύρου. Το σημείο $(0,0)$0,0 αντιστοιχεί στην άνω αριστερή γωνία της οθόνης. Η θετική φορά του οριζόντιου άξονα είναι προς τα δεξιά και του κατακόρυφου άξονα προς τα κάτω.

Οι διαστάσεις της επιφάνειας σχεδίασης καθορίζονται αρχικά με την εντολή ***glutInitWindowSize***

void glutInitWindowSize(int width, int height);

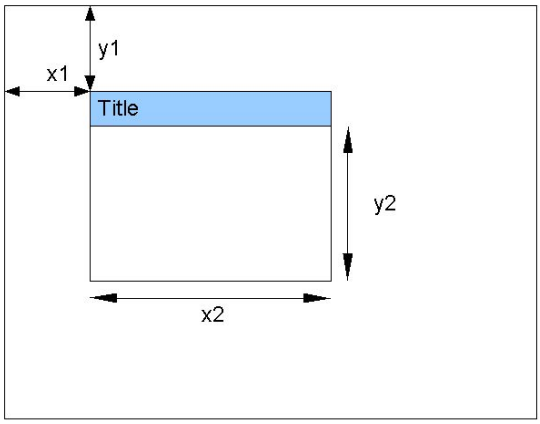
όπου *width* και *height* το πλάτος και ύψος της επιφάνειας σχεδίασης της εφαρμογής σε pixels. Επισημαίνουμε ότι οι διαστάσεις της επιφάνειας σχεδίασης μπορούν να μεταβληθούν από το χρήστη σε κάθε χρονική στιγμή κατά την εκτέλεση του προγράμματος με τη χρήση του ποντικιού.

Το παράθυρο σχηματίζεται δίνοντας την εντολή ***glutCreateWindow***:

int glutCreateWindow(const char *title);

η οποία επιστρέφει μια ακέραιη τιμή που λειτουργεί ως αναγνωριστικό του παραθύρου (χρήσιμη παράμετρος στην περίπτωση δημιουργίας πολλαπλών παραθύρων). Η παράμετρος *title* αντιστοιχεί στον τίτλο του παραθύρου.

Η επίδραση των εντολών ***glutInitWindowPosition*** και ***glutInitWindowSize*** και ***glutCreateWindow*** φαίνεται στο Σχήμα 2.1.



Σχ. 2.1: Παράμετροι σχηματισμού παραθύρου

Παράδειγμα: Αρχικοποίηση παραθύρου εφαρμογής και ρύθμιση των διαστάσεών του από τη γραμμή εντολών

```
#include <glut.h>
#include <stdlib.h>

int width;
int height;

void display()
{
	glClearColor(1,1,1,0);
	glClear(GL_COLOR_BUFFER_BIT);
	glFlush();
}

int main(int argc, char** argv)
{

	//The application demands two command-line arguments which
	//define the width and height of its window.

	//argv[0] returns the name of the application
	//argv[1] is the window width
	//argv[2] is the window height
	if (argc!=3)
```

```
	{
		printf("Bad arguments");
		exit(0);
	}

	//Converting command-line string arguments to integers
	// Converting ASCII characters to integers
	// Using function: int atoi(const char //*string)
	width=atoi(argv[1]);
	height=atoi(argv[2]);

	glutInit(&argc,argv);
	glutInitWindowPosition(50,50);
	glutInitWindowSize(width,height);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutCreateWindow("Window initialization");

	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(0,30,0,30);
	glutDisplayFunc(display);
	glutMainLoop();

	return 0;
}
```

2.2 Παράμετροι λειτουργίας παραθύρων

Κατά την αρχικοποίηση του παραθύρου ορίζουμε αν αυτό προορίζεται να περιέχει στατικές σκηνές ή κινούμενα γραφικά όπως και το χρωματικό μοντέλο που θα χρησιμοποιηθεί. Οι ρυθμίσεις αυτές δίνονται με την εντολή ***glutInitDisplayMode***:

void glutInitDisplayMode(unsigned int mode);

όπου *mode* προκαθορισμένες αριθμητικές σταθερές που παίρνουν τις παρακάτω τιμές:

GLUT_SINGLE: Για την απεικόνιση χρησιμοποιείται η τεχνική της απλής ενταμίευσης (χρησιμοποιείται ένας ενταμιευτής χρωματικών τιμών). Αυτή η ρύθμιση επιλέγεται για τη σχεδίαση στατικών σκηνών.
GLUT_DOUBLE: Χρησιμοποιείται η τεχνική της διπλής ενταμίευσης (double buffering) χρησιμοποιούνται δηλαδή δύο ενταμιευτές χρωματικών τιμών. Η επιλογή αυτή ενδείκνυται για την παρουσίαση κινούμενων γραφικών, για λόγους που θα εξηγηθούν στο Κεφάλαιο “Κινούμενα γραφικά”.
GLUT_RGB: Τα χρώματα ορίζονται με την περιγραφή τους στο χρωματικό μοντέλο RGB. Κάθε χρώμα δηλαδή περιγράφεται από τους συντελεστές βάρους της κόκκινης , της πράσινης και μπλε χρωματικής συνιστώσας του.
GLUT_RGBA: Τα χρώματα ορίζονται στο μοντέλο RGBA.
GLUT_INDEX: Χρήση του μοντέλου χρωματικών πινάκων (colour tables).

Οι παραπάνω σταθερές μπορούν να συνδυαστούν στο όρισμα της ***glutInitDisplayMode*** με τη χρήση τελεστών OR, υπό την προϋπόθεση βέβαια ότι εκφράζουν διαφορετικές ιδιότητες. Π.χ. για τη χρήση του χρωματικού μοντέλου RGB και την εφαρμογή διπλής ενταμίευσης, δίνουμε την εντολή

```
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
```

Στην περίπτωση που δεν δίνεται εντολή ***glutInitDisplayMode*** στον κώδικα, η OpenGL θεωρεί ως προκαθορισμένη κατάσταση τη χρήση του χρωματικού μοντέλου RGB και την τεχνική της απλής ενταμίευσης.

2.3 Γεγονότα

Ως γεγονός ορίζουμε την καταγραφή κάποιας δραστηριότητας του συστήματος, συνήθως μιας δραστηριότητας από κάποια συσκευή εισόδου όπως ένα πληκτρολόγιο ή ένα ποντίκι. Ωστόσο υπάρχουν και γεγονότα που εγείρονται από το λειτουργικό σύστημα υπό ορισμένες συνθήκες. Μία καταγραφή γεγονότος περιέχει πληροφορίες που το προσδιορίζουν επακριβώς. Π.χ. μία καταγραφή γεγονότος από το πληκτρολόγιο περιέχει την ταυτότητα του πλήκτρου που πιέστηκε και τη θέση του δείκτη όταν αυτό πατήθηκε. Μία καταγραφή γεγονότος από το ποντίκι περιέχει πληροφορίες για το πλήκτρο του ποντικιού που πιέστηκε και την τοποθεσία του δείκτη στην οθόνη όταν πιέστηκε.

Τα γεγονότα καταγράφονται σε μια σειρά , την οποία διαχειρίζεται το λειτουργικό σύστημα. Αυτό, τηρεί ένα αρχείο με όλα τα γεγονότα, παρακολουθεί τη σειρά με την οποία εγείρονται και αναθέτει τη διαχείριση κάθε γεγονότος σε ένα τμήμα κώδικα το οποίο έχει προκαθοριστεί να εκτελείται κατά την εμφάνιση του γεγονότος. Προφανώς, αυτό το προγραμματιστικό μοντέλο της ανταπόκρισης σε γεγονότα παρέχει τη δυνατότητα υλοποίησης αλληλεπιδραστικών εφαρμογών (event-driven programming).

2.4 Κατηγορίες γεγονότων στην OpenGL

Οι κατηγορίες γεγονότων που ορίζονται στην OpenGL είναι οι εξής:

α) Γεγονός σχεδίασης: Εμφανίζεται κάθε φορά που το σύστημα θα διαπιστώσει ότι απαιτείται σχεδίαση ή επανασχεδιασμός της σκηνής. Γεγονότα σχεδίασης εγείρονται:

- κατά τη μετακίνηση του παραθύρου στην οθόνη

- κατά την επαναφορά του παραθύρου της εφαρμογής στο προσκήνιο (restore)
- όταν δοθεί εντολή επανασχεδιασμού της σκηνής από τον προγραματιστή

Η τελευταία περίπτωση βρίσκει εφαρμογή όταν ο προγραμματιστής έχει τροποποιήσει τη σκηνή και θέλει να ανανεώσει το αποτέλεσμα και στην οθόνη. Στην περίπτωση αυτή, θα πρέπει να δοθεί εντολή (και συνεπώς να εμφανιστεί το γεγονός) σχεδιασμού της οθόνης. Ρητή εντολή επανασχεδιασμού της σκηνής από το χρήστη δίνεται χρησιμοποιώντας τη συνάρτηση ***glutPostRedisplay***:

void glutPostRedisplay( );

β) Γεγονός αλλαγής διαστάσεων παραθύρου (reshape): Εγείρεται την πρώτη φορά που σχεδιάζεται το παράθυρο (κατά την εκκίνηση της εφαρμογής) καθώς και κάθε φορά που ο χρήστης μεταβάλλει χειροκίνητα τις διαστάσεις του παραθύρου.

γ) Γεγονότα πληκτρολογίου (keyboard events): Εμφανίζονται κάθε φορά που πιέζουμε ένα από τα πλήκτρα του πληκτρολογίου. Δεδομένου ότι τα πλήκτρα διαχωρίζονται σε δύο κατηγορίες, τα **πλήκτρα χαρακτήρων** και τα **ειδικά πλήκτρα** (Function keys, Insert, Page Up, Page Down κλπ.), ομοίως ορίζονται και δύο κατηγορίες γεγονότων πληκτρολογίου.

δ) Γεγονότα ποντικιού: Εμφανίζονται κατά τη χρήση του ποντικιού. Στην κατηγορία αυτή ανήκει η πίεση των πλήκτρων του ποντικιού, η ενεργή κίνηση του δείκτη (κίνηση του δείκτη του ποντικιού με ένα από τα πλήκτρα του πιεσμένο) και η παθητική κίνηση (κίνηση του δείκτη του ποντικιού χωρίς πιεσμένο πλήκτρο).

ε) Γεγονότα χρονισμού: Στην κατηγορία αυτή ανήκουν γεγονότα που εγείρονται σε συγκεκριμένη χρονική στιγμή. Η έγερσή τους στη συγκεκριμένη αυτή στιγμή ενεργοποιεί την εκτέλεση κάποιου κώδικα. Πρακτικά, τα γεγονότα χρονισμού λειτουργούν ως μετρητές (timers) για τον προγραμματισμό εργασιών (scheduling).

στ) "Γεγονός αδρανείας": Υπάρχουν περιπτώσεις στις οποίες επιθυμούμε την επ' άπειρον εκτέλεση κάποιου τμήματος κώδικα, ανεξαρτήτως της εμφάνισης γεγονότων, όπως λ.χ. σε εφαρμογές κινούμενων γραφικών. Στην περίπτωση αυτή, θεωρούμε ότι αυτό το τμήμα κώδικα θα εκτελείται κατά την εμφάνιση ενός "γεγονότος" αδράνειας του συστήματος. Το σύστημα θεωρείται αδρανές όταν δεν εγείρονται γεγονότα.

2.5 Κύκλος διαχείρισης γεγονότων

Στην OpenGL, προκειμένου ένα πρόγραμμα να ανταποκρίνεται σε γεγονότα, θα πρέπει να ενεργοποιηθεί ο ***κύκλος διαχείρισης γεγονότων*** (event processing loop). Η ενεργοποίηση του κύκλου διαχείρισης γεγονότων γίνεται δίνοντας την εντολή **glutMainLoop**:

void glutMainLoop( );

Μετά την ενεργοποίηση του κύκλου διαχείρισης γεγονότων ***η εφαρμογή θα αναμένει επ' άπειρον και θα ανταποκρίνεται σε γεγονότα που θα εγερθούν και στα οποία θα έχει καταχωρηθεί κώδικας προς εκτέλεση.***

Ουσιαστικά, η ανταπόκριση μιας εφαρμογής σε γεγονότα συνίσταται στην ανάθεση ενός κώδικα προς εκτέλεση ανά είδος γεγονότος. Ο κώδικας που θα εκτελείται κατά την εμφάνιση κάθε γεγονότος ορίζεται στις **συναρτήσεις διαχείρισης γεγονότων** ή αλλιώς **συναρτήσεις κλήσης** (**callback functions**). Προφανώς, γεγονότα για οποία δεν έχει καταχωρηθεί συνάρτηση διαχείρισης δεν έχουν καμία επίδραση στην εκτέλεση του προγράμματος. Ο παρακάτω ψευδοκώδικας περιγράφει τον κύκλο διαχείρισης γεγονότων

```
while(1)  //Εκτέλεση βρόχου επ' άπειρον
{
        if (γεγονός σχεδιασμού/επανασχεδιασμού)
        { εκτέλεση κώδικα διαχείρισης γεγονότος σχεδιασμού/επανασχεδιασμού; }

        if (αλλαγή διαστάσεων παραθύρου)
        {εκτέλεση κώδικα διαχείρισης του γεγονότος αλλαγής διάστασης παραθύρου; }

        if (εμφάνιση γεγονότος πληκτρολογίου/ποντικιού)
        { εκτέλεση κώδικα διαχείρισης γεγονότων πληκτρολογίου/ποντικιού }

        ……..

        εκτέλεση κώδικα διαχείρισης "γεγονότος αδρανείας" ;
}
```

2.6 Συναρτήσεις διαχείρισης γεγονότων

Η αντιστοίχιση συναρτήσεων διαχείρισης σε κατηγορίες γεγονότων γίνεται με τη χρήση συγκεκριμένων εντολών. Ανάλογα με το γεγονός που θα καλούνται να διαχειριστούν, οι συναρτήσεις κλήσης θα πρέπει να έχουν και συγκεκριμένη δομή, σε ό,τι αφορά το πλήθος και τον τύπο των ορισμάτων τους. **Μέσω των ορισμάτων που ορίζονται σε κάθε συνάρτηση κλήσης, το λειτουργικό σύστημα παρέχει στον προγραμματιστή πληροφορίες που προσδιορίζουν το εκάστοτε γεγονός,** όπως πχ το χαρακτήρα του πλήκτρου που πιέστηκε ή τις συντεταγμένες της θέσης όπου πιέστηκε ένα πλήκτρο του ποντικιού. Επομένως ο προγραμματιστής έχει τη δυνατότητα να διαχειριστεί το κάθε γεγονός ανάλογα με τις τιμές που ανακτά.

Στη συνέχεια δίνονται ορισμένες εντολές που χρησιμοποιούνται για την καταχώρηση συναρτήσεων κλήσης σε συνήθεις τύπους γεγονότων.

1) void glutDisplayFunc(void display()):

Στην εντολή ***glutDisplayFunc*** δίνουμε ως όρισμα τη συνάρτηση, η οποία θα εκτελείται όποτε εγείρεται γεγονός απαίτησης σχεδιασμού/επανασχεδιασμού της σκηνής (έστω **display( )**). Ουσιαστικά, όλες οι ρουτίνες σχεδίασης που αναφέραμε στο Κεφάλαιο 1, καθώς και οι κλήσεις των display lists, συνήθως τοποθετούνται στο εσωτερικό της συνάρτησης display. Η συνάρτηση κλήσης display πρέπει να έχει συγκεκριμένη δομή: δεν επιστρέφει τιμή και δε δέχεται ορίσματα. Η display θα εκτελείται στις περιπτώσεις μετακίνησης του παραθύρου σχεδίασης, της επαναφοράς του στο προσκήνιο και στην περίπτωση που ο προγραμματιστής απαιτεί τον επενασχεδιασμό της σκηνής, χρησιμοποιώντας την εντολή ***glutPostRedisplay***.

Παράδειγμα: Διαχείριση γεγονότων σχεδιασμού

```
#include <glut.h>

GLsizei width=640;
GLsizei height=480;

GLuint x1=10,y1=10;
GLuint x2=20,y2=20;

void display()
{
      printf("Display event detected.\n");
      glClearColor(1,1,1,0);
      glClear(GL_COLOR_BUFFER_BIT);

      glColor3f(1,0,0);
```

```
	glRecti(x1,y1,x2,y2);
	glFlush();
}


int main(int argc, char** argv)
{
	glutInit(&argc,argv);
	glutInitWindowPosition(50,50);
	glutInitWindowSize(width,height);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutCreateWindow("Display events");

	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(0,30,0,30);
	glutDisplayFunc(display);
	glutMainLoop();

	return 0;
}
```

2) void glutReshapeFunc(void reshape(int width, int height))

Καταχωρεί τη συνάρτηση κλήσης ***reshape*** για τη διαχείριση γεγονότων γεγονός αλλαγής των διαστάσεων του παραθύρου. Η συνάρτηση reshape μέσω των ακεραίων ορισμάτων *width* και *height*, δίνει στον προγραμματιστή τις νέες διαστάσεις της επιφάνειας σχεδίασης (πλάτος και ύψος σε pixels).

Η συνάρτηση κλήσης reshape είναι χρήσιμη για να επαναπροσδιορίσουμε το εύρος της επιφάνειας σχεδίασης που θα αξιοποιηθεί μετά από αλλαγές των διαστάσεων του παραθύρου. Εάν δεν ορίσουμε συνάρτηση διαχείρισης του γεγονότος, η μηχανή της OpenGL αυτομάτως επεκτείνει τη σχεδίαση της σκηνής σε όλη τη διαθέσιμη επιφάνεια που προσφέρει το παράθυρο σχεδίασης. Αυτό σημαίνει ότι εάν θελήσουμε να επιβάλλουμε κάποιους περιορισμούς - όπως λ.χ. να χρησιμοποιήσουμε διαφορετική αναλογία πλάτους-ύψους από αυτή που έχει η επιφάνεια σχεδίασης στο σύνολό της – θα πρέπει να καταχωρήσουμε μια συνάρτηση κλήσης που ορίζει το παράθυρο σχεδίασης.

Παράδειγμα χρήσης της reshapeFunc θα παρουσιαστεί στη συνέχεια του κεφαλαίου, στην ενότητα “Παράθυρο παρατήρησης”.

3) void glutKeyboardFunc(void keyboard(unsigned char key, int x, int y)):

Καταχωρεί τη συνάρτηση κλήσης ***keyboard*** ως συνάρτηση διαχείρισης γεγονότων πληκτρολόγησης χαρακτήρων (character keys). Το όρισμα *key* (τύπου unsigned char) περιέχει τον χαρακτήρα που αντιστοιχεί στο πλήκτρο που πιέστηκε. Τα ορίσματα *x, y* προσδιορίζουν τη θέση στην επιφάνεια σχεδίασης, όπου βρισκόταν ο δείκτης του ποντικιού, όταν πιέστηκε το πλήκτρο. **Οι τιμές x,y ορίζονται**

σε σύστημα συντεταγμένων με ακέραιες τιμές συντεταγμένων, με αρχή την πάνω αριστερή γωνία της επιφάνειας σχεδίασης και θετική φορά στην οριζόντια και κατακόρυφη διεύθυνση προς τα δεξιά και κάτω αντίστοιχα.

Προφανώς, ανάλογα με την τιμή (είδος) του πλήκτρου που πιέστηκε, υπάρχει η δυνατότητα εκτέλεσης διαφορετικού κώδικα, χρησιμοποιώντας στο εσωτερικό της συνάρτησης keyboard δομές τύπου switch, if/else κλπ.

Παράδειγμα: Διαχείριση πληκτρολόγησης χαρακτήρων

```
#include <glut.h>

GLuint x1=10,y1=10;
GLuint x2=20,y2=20;

void display()
{
	glClearColor(1,1,1,0);
	glClear(GL_COLOR_BUFFER_BIT);

	glColor3f(1,0,0);
	glRecti(x1,y1,x2,y2);
	glFlush();
}

void keyboard(unsigned char key,int x, int y)
{
	printf("Keyboard event detected.\n");

	if (key=='t')
	{
		y1++;
		y2++;
	}
	if (key=='g')
	{
		y1--;
		y2--;
	}
	if (key=='f')
	{
		x1--;
		x2--;
	}
	if (key=='h')
	{
		x1++;
		x2++;
	}

	glutPostRedisplay();
}
```

```
int main(int argc, char** argv)
{
	glutInit(&argc,argv);
	glutInitWindowPosition(50,50);
	glutInitWindowSize(640,480);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutCreateWindow("Keyboard events");

	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(-5,35,-5,30);
	glutDisplayFunc(display);
	glutKeyboardFunc(keyboard);
	glutMainLoop();

	return 0;
}
```

4) void glutSpecialFunc(void specialKey(int key, int x, int y)):

Καταχωρεί τη συνάρτηση ***specialKey*** ως συνάρτηση διαχείρισης γεγονότων πληκτρολόγησης ειδικών πλήκτρων (special keys). Το όρισμα *key* (τύπου int) περιέχει την αριθμητική τιμή που καθορίζει το ειδικό πλήκτρο που πατήθηκε. Στην OpenGL, σε κάθε ειδικό πλήκτρο αντιστοιχεί μια συγκεκριμένη αριθμητική σταθερά, όπως φαίνεται στον Πίνακα 1. Τα ορίσματα *x*, *y* προσδιορίζουν τη θέση στην επιφάνεια σχεδίασης όπου βρισκόταν ο δείκτης του ποντικιού όταν πιέστηκε το πλήκτρο. **Οι συντεταγμένες *x*, *y* ορίζονται σε σύστημα με ακέραιες τιμές συντεταγμένων, με αρχή την πάνω αριστερή γωνία της επιφάνειας και θετική φορά στην οριζόντια και κατακόρυφη διεύθυνση προς τα δεξιά και κάτω αντίστοιχα**.

Όπως και στην προηγούμενη περίπτωση, ανάλογα με το πλήκτρο που πιέστηκε, υπάρχει η δυνατότητα εκτέλεσης διαφορετικού κώδικα.

Πίνακας 1: Κατηγορίες ειδικών πλήκτρων

Κατηγορία	**Σύμβολο στο πληκτρολόγιο**	**Αντίστοιχη αριθμητική σταθερά στην OpenGL**
Function keys	F1- F2 - … - F12	GLUT_KEY_F1 …. GLUT_KEY_F12
Πλήκτρα κατεύθυνσης (arrow keys)	↑, ↓, ←, →	GLUT_KEY_UP GLUT_KEY_DOWN GLUT_KEY_LEFT GLUT_KEY_RIGHT
Διάφορα πλήκτρα	Page Up Page Down Home End Insert	GLUT_KEY_PAGE_UP GLUT_KEY_PAGE_DOWN GLUT_KEY_HOME GLUT_KEY_END GLUT_KEY_INSERT

Παράδειγμα: Διαχείριση πληκτρολόγησης ειδικών πλήκτρων

```
#include <glut.h>

GLuint x1=10,y1=10;
GLuint x2=20,y2=20;

void display()
{
	glClearColor(1,1,1,0);
	glClear(GL_COLOR_BUFFER_BIT);

	glColor3f(1,0,0);
	glRecti(x1,y1,x2,y2);
	glFlush();
}

void specialKeys(int key,int x, int y)
{
	printf("Special key event detected.\n");

	if (key==GLUT_KEY_UP)
	{
		y1++;
		y2++;
	}
	if (key==GLUT_KEY_DOWN)
	{
		y1--;
		y2--;
	}
	if (key==GLUT_KEY_LEFT)
	{
```

```
            x1--;
            x2--;
      }
      if (key==GLUT_KEY_RIGHT)
      {
            x1++;
            x2++;
      }

      glutPostRedisplay();
}

int main(int argc, char** argv)
{
      glutInit(&argc,argv);
      glutInitWindowPosition(50,50);
      glutInitWindowSize(640,480);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
      glutCreateWindow("Special key events");

      glMatrixMode(GL_PROJECTION);
      gluOrtho2D(-5,35,-5,30);
      glutDisplayFunc(display);
      glutSpecialFunc(specialKeys);
      glutMainLoop();

      return 0;
}
```

5) void glutMouseFunc(void mouseClicked(int button, int state, int x, int y)):

Καταχωρεί τη συνάρτηση ***mouseClicked*** ως συνάρτηση διαχείρισης γεγονότων πίεσης πλήκτρων ποντικιού.

Η τιμή του ορίσματος *button* που επιστρέφει το λειτουργικό σύστημα εξαρτάται από το κουμπί του ποντικιού που πιέστηκε:
GL_LEFT_BUTTON: Πιέστηκε το αριστερό κουμπί του ποντικιού.
GL_MIDDLE_BUTTON: Πιέστηκε το μεσαίο κουμπί του ποντικιού.
GL_RIGHT_BUTTON: Πιέστηκε το το δεξί κουμπί του ποντικιού.

Η τιμή του ορίσματος *state* εξαρτάται από το αν το κουμπί του ποντικιού πιέστηκε ή απελευθερώθηκε (Η πίεση και η απελευθέρωση ενός κουμπιού του ποντικιού μπορούν να αντιμετωπιστούν ως διαφορετικά γεγονότα.):

GLUT_DOWN: Πίεση του πλήκτρου του ποντικιού.
GLUT_UP: Απελευθέρωση του πλήκτρου του ποντικιού.

Τα ορίσματα x, y εκφράζουν τη θέση του δείκτη του ποντικιού όταν πιέστηκε το κουμπί (ακέραιες τιμές με τη θέση $(0,0)$ στην πάνω αριστερή γωνία της επιφάνειας σχεδίασης της εφαρμογής).

Παράδειγμα: Διαχείριση mouse clicks

```
#include <glut.h>

//Clipping window limits
float xwmin=-50, xwmax=50, ywmin=-50, ywmax=50;

//Application window dimensions
int windowWidth=640, windowHeight=480;

GLfloat xw1,yw1,xw2,yw2;

void display()
{
	glClearColor(1,1,1,0);
	glClear(GL_COLOR_BUFFER_BIT);

	glRectf(xw1,yw1,xw2,yw2);

	glFlush();
}

void mouseButtonClicked(int button,int state,int x, int y)
{
	if(state==GLUT_DOWN)
	{
		printf("Mouse click event detected (button down).\n");

		//Converting coordinates captured by event handler to world
		coordinates
		xw1=(float)(xwmin+(float)(x)*(xwmax-xwmin)/(float)(windowWidth));
		yw1=(float)(ywmin+(float)(windowHeight-y)*(ywmax-
		ywmin)/(float)(windowHeight));

		xw2=xw1+20;
		yw2=yw1+20;

		if (button==GLUT_LEFT_BUTTON)
			glColor3f(1,0,0);
		else if (button==GLUT_MIDDLE_BUTTON)
			glColor3f(0,1,0);
		else if (button==GLUT_RIGHT_BUTTON)
			glColor3f(0,0,1);

		//Generate a display event
		glutPostRedisplay();
	}
}

int main(int argc, char** argv)
{
	glutInit(&argc,argv);
	glutInitWindowPosition(50,50);
```

```
	glutInitWindowSize(windowWidth,windowHeight);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutCreateWindow("Mouse click events");

	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(xwmin,xwmax,ywmin,ywmax);
	glutDisplayFunc(display);
	glutMouseFunc(mouseButtonClicked);
	glutMainLoop();

	return 0;
}
```

6)void glutMotionFunc(void activeMouseMotion(int x, int y)):

Η ***glutMotionFunc*** θέτει την ***activeMouseMotion*** ως συνάρτηση διαχείρισης του γεγονότος της ενεργής μετακίνησης του δείκτη του ποντικιού (μετακίνηση του ποντικιού με πιεσμένο πλήκτρο) Η συνάρτηση κλήσης περιέχει ως ορίσματα τις συντεταγμένες *x*, *y* (ορισμένες ως προς το σύστημα συντεταγμένων που υιοθετούν οι συναρτήσεις διαχείρισης γεγονότων) του σημείου όπου σύρεται ο δείκτης.

Παράδειγμα: Διαχείριση ενεργής κίνησης ποντικιού

```
#include <glut.h>

//Clipping window limits
float xwmin=-50, xwmax=50, ywmin=-50, ywmax=50;

//Application window dimensions
int windowWidth=640, windowHeight=480;

GLfloat xw1=0,yw1=0;
GLfloat xw2=xw1+20,yw2=yw1+20;

void display()
{
	glClearColor(1,1,1,0);
	glColor3f(0,0,1);
	glClear(GL_COLOR_BUFFER_BIT);

	glRectf(xw1,yw1,xw2,yw2);

	glFlush();
}

void mouseActiveMotion(int x, int y)
{
	printf("Mouse active motion detected.\n");

	xw1=(float)(xwmin+(float)(x)*(float)(xwmax-
xwmin)/(float)(windowWidth));
```

```
	yw1=(float)(ywmin+(float)(windowHeight-y)*(float)(ywmax-
ywmin)/(float)(windowHeight));

	xw2=xw1+20;
	yw2=yw1+20;

	//Generate a display event
	glutPostRedisplay();

}

int main(int argc, char** argv)
{
	glutInit(&argc,argv);
	glutInitWindowPosition(50,50);
	glutInitWindowSize(windowWidth,windowHeight);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutCreateWindow("Mouse active motion events");

	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(xwmin,xwmax,ywmin,ywmax);
	glutDisplayFunc(display);
	glutMotionFunc(mouseActiveMotion);
	glutMainLoop();

	return 0;
}
```

7) void glutPassiveMotionFunc(void passiveMouseMotion(int x, int y)):

Ομοίως με την προηγούμενη περίπτωση, με τη διαφορά ότι, η συνάρτηση εκτελείται κατά την παθητική μετακίνηση του δείκτη του ποντικιού (κίνηση ποντικιού χωρίς πιεσμένο πλήκτρο).

Παράδειγμα: Διαχείριση παθητικής κίνησης ποντικιού

```
#include <glut.h>

int windowWidth=640, windowHeight=480;
int xwmin=-50;
int ywmin=-50;
int xwmax=50;
int ywmax=50;

GLfloat xw1=10,yw1=10;
GLfloat xw2=xw1+100,yw2=yw1+100;

void display()
{
	glClearColor(1,1,1,0);
	glColor3f(0,0,1);
	glClear(GL_COLOR_BUFFER_BIT);
```

```
	glRectf(xw1,yw1,xw2,yw2);

	glFlush();
}

void mousePassiveMotion(int x, int y)
{
	printf("Mouse passive motion detected.\n");

	//Converting device coordinates to world coordinates

	xw1=(float)(xwmin+(float)(x)*(float)(xwmax-
xwmin)/(float)(windowWidth));
	yw1=(float)(ywmin+(float)(windowHeight-y)*(float)(ywmax-
ywmin)/(float)(windowHeight));

	xw2=xw1+20;
	yw2=yw1+20;

	glutPostRedisplay();

}

int main(int argc, char** argv)
{
	glutInit(&argc,argv);
	glutInitWindowPosition(50,50);
	glutInitWindowSize(640,480);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutCreateWindow("Mouse passive motion events");

	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(xwmin,xwmax,ywmin,ywmax);
	glutDisplayFunc(display);
	glutPassiveMotionFunc(mousePassiveMotion);
	glutMainLoop();

	return 0;
}
```

8) void glutIdleFunc(void animate( )):

Καταχωρεί τη συνάρτηση ***animate*** ως συνάρτηση διαχείρισης του "γεγονότος αδράνειας". Ουσιαστικά η συνάρτηση ***animate*** θα εκτελείται επαναληπτικά σε κάθε κύκλο διαχείρισης γεγονότων. Είναι ιδιαίτερα χρήσιμη σε περιπτώσεις που εκτελούμε συνεχείς μεταβολές και επιθυμούμε τακτικό επανασχεδιασμό της σκηνής, όπως στην περίπτωση των κινούμενων γραφικών (animation). Η συνάρτηση ***animate*** δε δέχεται ορίσματα και δεν επιστρέφει τιμή.

Παράδειγμα:

```
#include <glut.h>

GLuint x1=10;
GLuint y1=10;
GLuint x2=x1+100;
GLuint y2=y1+100;

void display()
{
	glClearColor(1,1,1,0);
	glClear(GL_COLOR_BUFFER_BIT);

	glColor3f(1,0,0);
	glRecti(x1,y1,x2,y2);
	glFlush();
}

void idle()
{
	printf("Idle event detected.\n");
	if (x2>640)
	{
		x1=0;
		x2=x1+100;
	}
	else
	{
		x1++;
		x2++;
	}
	glutPostRedisplay();
}

int main(int argc, char** argv)
{
	glutInit(&argc,argv);
	glutInitWindowPosition(50,50);
	glutInitWindowSize(640,480);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutCreateWindow("Idle events (A low-quality animation)");

	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(0,640,0,480);
	glutDisplayFunc(display);

	//Register an idle callback function.
	glutIdleFunc(idle);
	glutMainLoop();

	return 0;
}
```

9) void glutTimerFunc(unsigned int execTime, void timed(int val), value):

Καταχωρεί ως συνάρτηση κλήσης την ***timed*** στο γεγονός χρονισμένης εκτέλεσης. Αυτό σημαίνει ότι η συνάρτηση ***timed*** θα εκτελεστεί μία μόνο φορά και σε προγραμματισμένο χρονικό διάστημα τουλάχιστον *execTime* msec από τη στιγμή της έναρξης του κύκλου διαχείρισης γεγονότων. Η τιμή value δίνεται ως όρισμα στην συνάρτηση timed.

Παράδειγμα: Προγραμματισμός εκτέλεσης συνάρτησης

```
#include <glut.h>

GLuint x1=10;
GLuint y1=10;
GLuint x2=x1+100;
GLuint y2=y1+100;

void display()
{
	glClearColor(1,1,1,0);
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1,0,0);
	glRecti(x1,y1,x2,y2);
	glFlush();
}

void timerFunction(int value)
{
	printf("Timer event detected.\nValuePassed: %d\n",value);
	x1+=100;
	x2+=100;
	y1+=100;
	y2+=100;
	glutPostRedisplay();
}

int main(int argc, char** argv)
{
	glutInit(&argc,argv);
	glutInitWindowPosition(50,50);
	glutInitWindowSize(640,480);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutCreateWindow("Timer event");

	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(0,640,0,480);
	glutDisplayFunc(display);

	//Register a timer callback function.
	//It will be executed in 5 seconds. and passes the value of //10 to the
	timer function
	glutTimerFunc(5000,timerFunction,10);
	glutMainLoop();

	return 0;
}
```

Παρατήρηση:

Θα πρέπει να έχουμε υπόψη ότι, εάν χρησιμοποιούμε μια συνάρτηση κλήσης για να τροποποιήσουμε το περιεχόμενο της σκηνής, αυτό δε συνεπάγεται ότι με το πέρας της εκτέλεσης της συνάρτησης κλήσης, οι αλλαγές θα εμφανιστούν και στην οθόνη. Στην περίπτωση αυτή, θα πρέπει επιπρόσθετα να παραγάγουμε ένα γεγονός απαίτησης επανασχεδιασμού δίνοντας την εντολή ***glutPostRedisplay*** μέσα στον κώδικα της εκάστοτε συνάρτησης κλήσης.

2.7 Απεικονίσεις στις 2 διαστάσεις - Συντεταγμένες σκηνής - Συντεταγμένες συσκευής

Όταν δηλώνουμε τη θέση ενός σημείου χρησιμοποιώντας τις εντολές σχεδίασης ***glVertex*** που περιγράψαμε στο Κεφάλαιο 1, οι συντεταγμένες που αποδίδουμε στα σημεία, δεν αντιστοιχούν στις θέσεις των pixels στην οθόνη. Αντίθετα ορίζονται σε ένα σύστημα με απεριόριστο εύρος και αναλυτικότητα στις τιμές του, το σύστημα **συντεταγμένων σκηνής** (world coordinate system). Πρακτικά, το εύρος στο οποίο εκτείνονται οι τιμές των δηλούμενων σημείων επιλέγεται αυθαίρετα από τον εκάστοτε προγραμματιστή. Συνήθως επιλέγεται ένα εύρος τιμών που επιτρέπει μια "βολική" από πλευράς αναλυτικότητας απεικόνιση στην εκάστοτε περίπτωση.

Ωστόσο, όταν τα δηλούμενα σημεία προωθούνται προς απεικόνιση στην οθόνη, λόγω της πεπερασμένης αναλυτικότητάς της τελευταίας, για κάθε σχεδιαζόμενο σημείο, θα πρέπει να υπολογιστούν οι ακέραιες συντεταγμένες που καθορίζουν την αντίστοιχή του θέση στην επιφάνεια σχεδίασης. Υπολογίζουμε δηλαδη τις λεγόμενες **συντεταγμένες συσκευής**.

Η διαδικασία της εύρεσης των συντεταμένων συσκευής που αντιστοιχούν σε κάθε σημείο (το οποίο ορίζεται σε συντεταγμένες σκηνής) χωρίζεται σε δύο στάδια. Στο στάδιο της **αποκοπής** και στο στάδιο του **μετασχηματισμού παρατήρησης**. Στη συνέχεια του Κεφαλαίου θα αναλύσουμε την αποκοπή και το μετασχηματισμό παρατήρησης στις δύο διαστάσεις.

2.8 Αποκοπή στις δύο διαστάσεις

Με τον όρο αποκοπή εννοούμε τη διαδικασία κατά την οποία απομονώνουμε ένα τμήμα της σκηνικού για την αναπαράστασή του στη συσκευή εξόδου. Ουσιαστικά η απομόνωση αυτή στις δύο διαστάσεις επιτελείται ορίζοντας ως προς το σύστημα συντεταγμένων σκηνής τα όρια του **παραθύρου αποκοπής** (clipping window). Με τον όρο παράθυρο αποκοπής εννοούμε ένα ορθογώνιο, μέσα στο οποίο

περικλείεται το τμήμα της σκηνής που θέλουμε να απομονώσουμε. Ο καθορισμός του παραθύρου αποκοπής σε δύο διαστάσεις γίνεται χρησιμοποιώντας την εντολή ***gluOrtho2D*** της βιβλιοθήκης GLU:

void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);

όπου *left*, *right*, *bottom* και *top* το αριστερό, δεξιό, κάτω και άνω όριο του παραθύρου αποκοπής. Τα όρια του παραθύρου αποκοπής ορίζονται στο σύστημα συντεταγμένων σκηνής.

Όπως θα δούμε στο κεφάλαιο "Προβολές", ο καθορισμός του παραθύρου αποκοπής είναι μια διαδικασία που συνδέεται με τον ορισμό του μητρώου προβολής. Ουσιαστικά, με τη χρήση της εντολής ***gluOrtho2D***, ορίζουμε ταυτόχρονα ένα παράθυρο αποκοπής και την επιβολή παράλληλης προβολής στο επίπεδο XY. (περισσότερα για το συσχετισμό αποκοπής και προβολής στο Κεφάλαιο «Προβολές».) Συνεπώς πριν τον ορισμό του παραθύρου αποκοπής πρέπει να μεταβούμε στην κατάσταση επεξεργασίας του μητρώου προβολής. Η μετάβαση αυτή γίνεται με την έντολή ***glMatrixMode*** :

void glMatrixMode(GLenum mode);

Μεταβαίνουμε στην κατάσταση επεξεργασίας του μητρώου προβολής δίνοντας την παράμετρο *GL_PROJECTION*.

glMatrixMode(GL_PROJECTION);

Εαν στην εφαρμογή μας δεν καθοριστεί παράθυρο αποκοπής, η OpenGL θεωρεί προκαθορισμένο παράθυρο αποκοπής που εκτείνεται μεταξύ των ορίων $[-1,1]$ και στις δύο διαστάσεις (κανονικοποιημένο τετράγωνο).

2.9 Παράθυρο παρατήρησης

Στην OpenGL, αντί να χρησιμοποιήσουμε ολόκληρη τη διαθέσιμη επιφάνεια σχεδίασης της συσκευής εξόδου για την αναπαράσταση μιας σκηνής, μπορούμε να καθορίσουμε το τμήμα της επιφάνειας σχεδίασης (τα όρια των ακεραίων συντεταγμένων συσκευής $[xu_{\min}, xu_{\max}]$ και $[yu_{\min}, yu_{\max}]$) μέσα στο οποίο θέλουμε να σχεδιαστούν τα περιεχόμενα του παραθύρου αποκοπής Ο καθορισμός του εύρους των συντεταγμένων συσκευής που θα διατεθούν για τη σχεδιάση της αποκομμένης σκηνής γίνεται με τον καθορισμό ενός **παραθύρου παρατήρησης** (viewport). Ο καθορισμός των διαστάσεων ενός παραθύρου παρατήρησης γίνεται χρησιμοποιώντας την εντολή ***glViewport***:

void glViewport(GLint xmin, GLint ymin, GLsizei width, GLsizei height);

όπου οι παράμετροι *xmin* και *ymin* δηλώνουν το κάτω αριστερό άκρο του παραθύρου παρατήρησης. Οι παράμετροι *width*, *height* δηλώνουν το πλάτος και ύψος που θα καταλάβει στην επιφάνεια σχεδίασης το παράθυρο παρατήρησης.

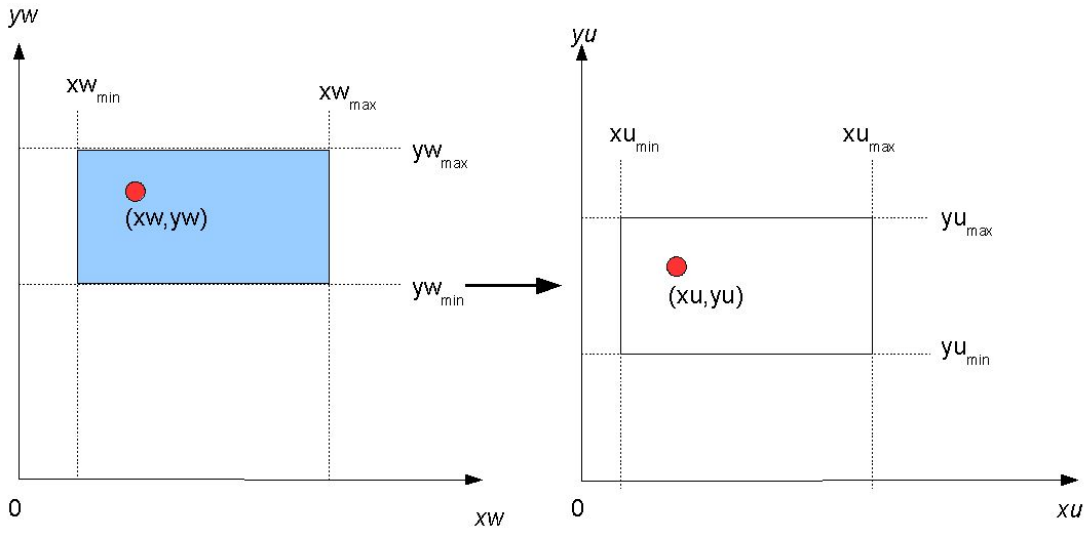
Η ***glViewport*** καθορίζει τη θέση του παραθύρου παρατήρησης στην επιφάνεια σχεδίασης και δηλώνεται πάντοτε μέσα στον κώδικα της συνάρτησης κλήσης που διαχειρίζεται το γεγονός reshape (και που καταχωρείται στην ***glutReshapeFunc***). Οι συντεταγμένες που δίνουμε ως ορίσματα στην ***glViewPort*** είναι ακέραιες **συντεταγμένες συσκευής**.

Σε περίπτωση που δεν χρησιμοποιείται η εντολή ***glViewport*** στο πρόγραμμα, η OpenGL δημιουργεί ένα προκαθορισμένο παράθυρο παρατήρησης που εκτείνεται σε ολόκληρη τη διαθέσιμη επιφάνεια σχεδίασης. Σε περίπτωση χειροκίνητης αλλαγής των διαστάσεων της επιφάνειας σχεδίασης (reshape), οι διαστάσεις του προκαθορισμένου παραθύρου παρατήρησης αναπροσαρμόζονται, ούτως ώστε αυτό να εξακολουθεί καταλαμβάνει ολόκληρη την επιφάνεια σχεδίασης .

2.10 Μετασχηματισμός παρατήρησης

Θεωρούμε ένα παράθυρο αποκοπής που εκτείνεται μεταξύ των τιμών $[xw_{\min}, xw_{\max}]$ και $[yw_{\min}, yw_{\max}]$ στο σύστημα συντεταγμένων σκηνής. Επίσης, θεωρούμε στην επιφάνεια σχεδίασης της συσκευής εξόδου ένα διαθέσιμο εύρος τιμών (συντεταγμένες συσκευής) που εκτείνεται μεταξύ των ορίων $[xu_{\min}, xu_{\max}]$ και $[yu_{\min}, yu_{\max}]$ Αναζητούμε το μετασχηματισμό που αντιστοιχίζει το παραπάνω εύρος συντεταγμένων σκηνής στο διαθέσιμο εύρος συντεταγμένων συσκευής.

Δεδομένου ότι οι αναλογίες μεταξύ των αποστάσεων ενός σημείου από τα άκρα των παραθύρων διατηρούνται, έχουμε



$$\frac{xw - xw_{\min}}{xw_{\max} - xw_{\min}} = \frac{xu - xu_{\min}}{xu_{\max} - xu_{\min}}$$

$$\frac{yw - yw_{\min}}{yw_{\max} - yw_{\min}} = \frac{yu - yu_{\min}}{yu_{\max} - yu_{\min}}$$

οπότε επιλύοντας ως προς x_u και y_u προκύπτει

$$xu = s_x \cdot xw + t_x$$
$$yu = s_y \cdot yw + t_y$$

με τις παραμέτρους s_x, s_y, t_x και t_y να προκύπτουν από τις σχέσεις:

$$s_x = \frac{xu_{\max} - xu_{\min}}{xw_{\max} - xw_{\min}}$$
$$s_y = \frac{yu_{\max} - yu_{\min}}{yw_{\max} - yw_{\min}}$$
$$t_x = \frac{xu_{\min} \cdot xw_{\max} - xu_{\max} \cdot xw_{\min}}{xw_{\max} - xw_{\min}}$$
$$t_y = \frac{yu_{\min} \cdot yw_{\max} - yu_{\max} \cdot yw_{\min}}{yw_{\max} - yw_{\min}}$$

και σε μορφή μητρώου έχουμε

$$\begin{bmatrix} x_v \\ y_v \\ 1 \end{bmatrix} = M_{w,v} \cdot \begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix}$$

με

$$M_{w,v} = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

2.11 Κανονικοποίηση συντεταγμένων

Στην OpenGL, μετά τη διαδικασία της αποκοπής και πριν την απόδοση συντεταγμένων συσκευής, εκτελείται μια διαδικασία κανονικοποίησης των αποκομμένων συντεταγμένων σε ένα προκαθορισμένο εύρος τιμών. Αυτό γίνεται ούτως ώστε η αναπαράσταση των συντεταγμένων των απομονωμένων σχημάτων να είναι ανεξάρτητη της χρησιμοποιούμενης ανάλυσης της συσκευής εξόδου. Οι συντεταγμένες των σχημάτων που περιέχονται στο παράθυρο αποκοπής κανονικοποιούνται στο εύρος τιμών $[-1,1]$ (κανονικοποιημένο τετράγωνο). Προφανώς, το μητρώο του μετασχηματισμού κανονικοποίησης προκύπτει από τους τύπους μετασχηματισμού παρατήρησης, εάν θέσουμε:

$$x_{u\min} = y_{u\min} = -1$$
$$x_{u\max} = y_{u\max} = 1$$

οπότε προκύπτει το μητρώο μετασχηματισμού παρατήρησης

$$M_{w,ns} = \begin{bmatrix} \dfrac{2}{x_{w\max} - x_{w\min}} & 0 & -\dfrac{x_{w\min} + x_{w\max}}{x_{w\max} - x_{w\min}} \\ 0 & \dfrac{2}{y_{w\max} - y_{w\min}} & -\dfrac{y_{w\min} + y_{w\max}}{y_{w\max} - y_{w\min}} \\ 0 & 0 & 1 \end{bmatrix}$$

Κατόπιν, οι κανονικοποιημένες συντεταγμένες, μετατρέπονται σε συντεταγμένες συσκευής βάσει του μητρώου μετασχηματισμού:

$$M_{ns,v} = \begin{bmatrix} \dfrac{x_{u\max} - x_{u\min}}{2} & 0 & \dfrac{x_{u\min} + x_{u\max}}{2} \\ 0 & \dfrac{y_{u\max} - y_{u\min}}{2} & \dfrac{y_{u\min} + y_{u\max}}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

Παράδειγμα: Καθορισμός τετραγώνου παραθύρου παρατήρησης σε επιφάνεια αυθαίρετης αναλογίας πλάτους-ύψους

```
#include <glut.h>

GLsizei width=640;
GLsizei height=480;

GLuint x1=10,y1=10;
GLuint x2=20,y2=20;

void display()
{
	printf("Display event detected.\n");
	glClearColor(1,1,1,0);
	glClear(GL_COLOR_BUFFER_BIT);

	glColor3f(1,0,0);
	glRecti(x1,y1,x2,y2);
	glFlush();
}

void reshape(int newWidth,int newHeight)
{
	printf("Reshape event detected.\n");

	//Defining a square viewport
	if (newWidth<newHeight)
	{
		width=newWidth;
		height=width;
	}
	else
	{
		height=newHeight;
		width=height;
	}
	glViewport(0,0,width,height);
}

int main(int argc, char** argv)
{
	glutInit(&argc,argv);
	glutInitWindowPosition(50,50);
	glutInitWindowSize(width,height);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutCreateWindow("Reshape events");

	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(0,30,0,30);
	glutDisplayFunc(display);
	glutReshapeFunc(reshape);
	glutMainLoop();

	return 0;
}
```

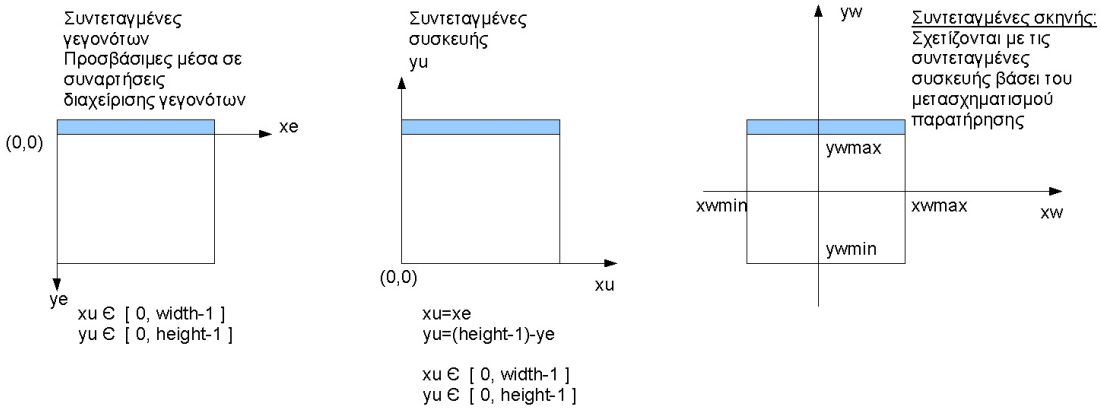
ΠΡΟΣΟΧΗ!!!

Στην περίπτωση διαχείρισης γεγονότων:

Αν θεωρήσουμε ένα pixel της επιφάνειας σχεδίασης, **η έκφρασή του με συντεταγμένες συσκευής διαφέρει από την εκφρασή του σε συντεταγμένες που λαμβάνουμε από τα ορίσματα των συναρτήσεων κλήσης,** αν κάνουμε λ.χ. κλικ σε αυτό.

Συγκεκριμένα, **οι συντεταγμένες-ορίσματα που λαμβάνουμε από τις συναρτήσεις κλήσης ορίζονται θεωρώντας την αρχή των αξόνων στην πάνω αριστερή γωνία της επιφάνειας σχεδίασης. Το ίδιο σημείο εκφράζεται με διαφορετικές τιμές στο σύστημα συντεταγμένων συσκευής, λόγω της διαφοράς σύμβασης ως προς την αρχή των αξόνων.** Δηλαδή, εάν κάνουμε click με το ποντίκι σε ένα σημείο που η συνάρτηση κλήσης θα χαρακτηρίσει με το ζεύγος τιμών (x,y) ,το σημείο αυτό, στο σύστημα συντεταγμένων συσκευής, θα εκφράζεται με τις τιμές (x, height-y).

Επίσης στην περίπτωση που θέλουμε, κάνοντας κλικ σε ένα pixel, να σχεδιάσουμε ένα σημείο στη θέση που θα δοθεί, τότε, στην αντίστοιχη εντολή σχεδίασης, **το σημείο θα πρέπει να εκφράστεί ως προς το σύστημα συντεταγμένων σκηνής**. Θα πρέπει δηλαδή να βρούμε οι συντεταγμένες συσκευής του pixel που επιλέξαμε σε ποιες συντεταγμένες σκηνής αντιστοιχούν. Η αντιστοιχία αυτή υπολογίζεται αντιστρέφοντας το μετασχηματισμό παρατήρησης που προαναφέρθηκε.



2.12 Επιδράσεις των παραθύρων αποκοπής και παρατήρησης

Στο σημείο αυτό είναι χρήσιμο να επισημάνουμε και να διαχωρίσουμε το ρόλο που παίζουν το παράθυρο αποκοπής και το παράθυρο παρατήρησης.

Το παράθυρο αποκοπής καθορίζει **ποιο τμήμα της σκηνής επιθυμούμε θα απομονώσουμε**, ενώ το παράθυρο παρατήρησης καθορίζει **το εύρος της επιφάνειας στο οποίο θα σχεδιαστεί το αποκομμένο τμήμα της σκηνής**.

Μεταβάλλοντας το έυρος του παραθύρου αποκοπής μπορούμε να δημιουργήσουμε εφφέ μεγέθυνσης ή σμίκρυνσης (zoom in – zoom out). Μεταβάλλοντας τη σχετική αναλογία πλάτους ύψους των παραθύρων αποκοπής και παρατήρησης επιτυγχάνουμε ή συμπίεση ή το τέντωμα της σκηνής ως προς μία από τις δύο διαστάσεις (stretching/skewing).

Η επιλογή των διαστάσεων των παραθύρων αποκοπής και παρατήρησης είναι σημαντική παράμετρος, όταν ο προγραμματιστής θέλει να αποδώσει το αποκομμένο τμήμα της σκηνής χωρίς ανεπιθύμητες παραμορφώσεις. Στην περίπτωση αυτή, θα πρέπει να ρυθμίσει προσεκτικά την αναλογία πλάτους-ύψους (aspect ratio) του παραθύρου αποκοπής και του παραθύρου παρατήρησης. Στην περίπτωση λ.χ. που ορίζουμε ένα κύκλο, προκειμένου να σχεδιαστεί στην οθόνη ως κύκλος, θα πρέπει η αναλογία πλάτους-ύψους και των δύο παραθύρων να είναι η ίδια . Στην αντίθετη περίπτωση, θα σχεδιαστεί έλλειψη.

Είναι επίσης σημαντικό να κατανοήσουμε ως προς ποιο σύστημα συντεταγμένων ορίζεται το κάθε παράθυρο. Συγκεκριμένα:

- Το παράθυρο αποκοπής ορίζεται σε *συντεταγμένες σκηνής*.
- Το παράθυρο παρατήρησης ορίζεται σε *συντεταγμένες συσκευής*.

Οι διαφορές μεταξύ των παραθύρων αποκοπής και παρατήρησης συνοψίζονται στον ακόλουθο πίνακα.

	Παράθυρο αποκοπής (gluOrtho2D)	**Παράθυρο παρατήρησης (glViewport)**
Καθορίζει:	τι θα σχεδιαστεί $[xw_{\min}, xw_{\max}], [yw_{\min}, yw_{\max}]$	πού θα σχεδιαστεί $[xu_{\min}, xu_{\max}], [yu_{\min}, yu_{\max}]$
Ορίζεται σε:	συντεταγμένες σκηνής	συντεταγμένες συσκευής