# frmMenu:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

public partial class frmMenu : Form

{

int size; //Antall brikker langs hver side i spillet

string path = ""; //Filplassering til bilde

public frmMenu()

{

InitializeComponent();

}

private void btnSelectImage\_Click(object sender, EventArgs e) // Metoden er lånt av Arne Wallerud. Den viser en dialogboks der man kan veget et bilde man vil bruke i spillet.

{

{

OpenFileDialog filePathDialog = new OpenFileDialog(); //Lager en ny "OpenFileDialog"

filePathDialog.Filter = "All Graphics Types|\*.bmp;\*.jpg;\*.jpeg;\*.png;\*.tif;\*.tiff" +

"BMP|\*.bmp|GIF|\*.gif|JPG|\*.jpg;\*.jpeg|PNG|\*.png|TIFF|\*.tif;\*.tiff|"; // Angi filtyper det skal være mulig å velge.

filePathDialog.Title = "Velg bilde"; //Legger til overskrift i FileDialog-vinduet

if (filePathDialog.ShowDialog() == DialogResult.OK) //Viser filePathDialog-dialogboksen. Hvis lastingen av bilde var vellykket:

{

try //(prøv) Gjør dette:

{

path = filePathDialog.FileName; //Legg inn filepath til det valgte bildet i variabelen path.

pic1.Image = Image.FromFile(path); //Legg det valgte bildet inn i den lille pictureboksen.(forhåndsvisning)

}

catch (Exception) //Hvis dette ikke går:

{

MessageBox.Show("Feil under lasting av bilde"); //Vis en messagebox med meldingen: "Feil under lasting av bilde".

throw;

//Dette gjør at man ikke får en feilmelding som stopper programmet.

}

}

}

}

private void btnStart\_Click(object sender, EventArgs e) //Metoden åpner et nytt vindu og sender med spillstørrelse, koordinater og filplassering.

{

size = Convert.ToInt32(numSize.Text);

int size0 = size - 1; // size0 er antall brikker langs hver side - 1. Det må trekkes fra en, fordi 0 også er en verdi.

Point missingPiece; //Koordinatene til den manglende brikken.

switch (cmbMissing.Text) //Velger koordinater til den manglende brikken. Det sjekkes hvilken tekst som er valgt, og det gis derretter en verdi til variabelen missingpiece av datatypen point, som brukes som koordinater for den manglende brikken. Siden det bare kan velges hjørner eller brikken i midten, trenger koorinatene bare å inneholde tallene 0, size0 og size0/2.

{

case "Senter":

missingPiece = new Point(size0 / 2, size0 / 2);

break;

case "Nede til høyre":

missingPiece = new Point(size0, size0);

break;

case "Nede til venstre":

missingPiece = new Point(0, size0);

break;

case "Oppe til høyre":

missingPiece = new Point(size0, 0);

break;

case "Oppe til venstre":

default:

missingPiece = new Point(0, 0);

break;

}

game frmgame = new game(size, missingPiece, path); //Starter "game" og sender med verdier til constructor-metoden.

frmgame.ShowDialog(); // Viser det nye vinduet.

}

}

}

# frmGame

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

public partial class frmGame : Form

{

string[] pieceNames; //Array der navn på brikkene lagres.

string[] pieceFasit; //Array der brikkenavnene lagres i riktig rekkefølge.

Point[] pieceCoordinates; //Array av point for hver brikke i spillet.

int piecesPerSide; //Antall brikker langs hver side i spillet.

int pieceSize; //størrelsen på hver enkelt brikke.

int topSpace; //Mellomrom mellom toppen av vinduet og spillbrikkene.

int counter = 0; //variabel som lagrer antall trekk.

byte gamestate = 0; //Statusen spillet er i. (ustartet, startet, pause eller ferdig.)

bool eventhandlerstate = false; //Variabel som holder kontroll på om eventhandlerene er fjernet eller ikke.

Random r = new Random(); // Tilfeldig tall.

Bitmap usedPicture; //Bildet som brukes.

TimeSpan clock; //Tid som er brukt.

public frmGame(int size, Point missingPiece, string path)

{

InitializeComponent();

CreatePieces(size, missingPiece, path);

this.Text = "Skyvepuslespill " + size + " x " + size; //Legger til navn på vinduet.

txtCounter.Text = "0"; //Skriver 0 i txtCounter textboksen.

}

private void CreatePieces(int size, Point missingPiece, string path) //Metode som lager brikkene i spillet.

{

piecesPerSide = size; //Antall brikker langs hver side.

CalculateSizeAndPositions(); //Regner ut størrelse og plassering til andre elementer i vinduet.

pieceNames = new string[piecesPerSide \* piecesPerSide]; //Definerer hvor mange verdier arrayet skal bestå av.

pieceCoordinates = new Point[piecesPerSide \* piecesPerSide]; //Definerer hvor mange verdier arrayet skal bestå av.

for (int x = 0; x < (piecesPerSide); x++)

{

for (int y = 0; y < (piecesPerSide); y++) //Nested for-loop for å gå gjennom alle koordinatene.

{

int pieceNumber = y \* piecesPerSide + x; //pieceNumber er nummeret i rekken på den aktuelle brikken.

string pieceName = "btn" + y + "" + x; //pieceName er navnet til den aktuelle brikken.

if (missingPiece.X == x && missingPiece.Y == y) //sjekker om de aktuelle koordinatene er koordinatene til tomrommet.

{

pieceNames[pieceNumber] = "empty"; //Legger navnet til tomrommet inn i et array.

pieceCoordinates[pieceNumber] = new Point(x, y); //legger koordinatene til tomrommet inn i et array.

}

else //Lag brikke.

{

pieceNames[pieceNumber] = pieceName; //Legger navnet til boksen inn i et array.

pieceCoordinates[pieceNumber] = new Point(x, y); //legger koordinatene til boksen inn i et array//Lager Picturebox

PictureBox newPiece = new PictureBox(); //Lager en ny picturebox

newPiece.Width = pieceSize; //Legger inn bredde og høyde.

newPiece.Height = pieceSize;

newPiece.Location = NameToRealCoordinates(pieceName); // Legger til de reelle koordinatene til brikken.

if (path.Length == 0) //Hvis det ikke er lagt inn en "filepath", skal standardbildet brukes.

{

usedPicture = new Bitmap(Properties.Resources.tree\_square);

}

else //ellers skal det angitte bilde brukes.

{

usedPicture = new Bitmap(path);

}

Bitmap imagePart = CropImage(piecesPerSide, usedPicture, x, y); //Metoden beskjærer bildet

newPiece.Image = imagePart; //Legger inn bildet

newPiece.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage; //Bildemodusen i texstboksen settes til "stretch image".

newPiece.Name = pieceName; //Pictureboxen får navn.

this.Controls.Add(newPiece); //Picturebox blir blir lagt til i bildet.

}

}

}

pieceFasit = new string[piecesPerSide\*piecesPerSide]; //Størrelsen på arrayet blir angitt.

for (int i = 0; i <= pieceNames.GetUpperBound(0); i++) //Arrayet pieceFasit, settes lik arrayet pieceNames.

{

pieceFasit[i] = pieceNames[i];

}

}

private void gameButton\_Click(object sender, EventArgs e) //Eventhandleren kjøres når en brikke blir klikket på.

{

counter++; //Teller antall klikk

txtCounter.Text = counter.ToString(); //Viser antall klikk på skjerm.

PictureBox piecePressed = (PictureBox)sender; //konverterer "sender"(knappen som ble trykket) til piecePressed.

int index = Array.IndexOf(pieceNames, piecePressed.Name); //Finner posisjonen(i arrayet) til brikken som ble trykket på.

Point PieceClicked;

PieceClicked = pieceCoordinates[index]; //Velger ut riktig point fra arrayet

//Multipliserer ut koordinatene, og man får dermed posisjonen til knappen som ble trykket på

RearrangePieces(PieceClicked); //Metoden flytter brikkene i arrayet "pieceNames".

UpdatePositions(pieceSize / 16); //Metoden oppdaterer den virkelige posisjonen til brikkene.

bool equal = true; //Variabel som skal vise om spillet er løst eller ikke.

string string1;

string string2;

for (int j = 0; j <= pieceNames.GetUpperBound(0); j++) //Looper gjennom alle brikkenavnene for å sjekke at de ikke er på rett plass.

{

string1 = pieceNames[j]; //brikkenavn fra "pieceNames".

string2 = pieceFasit[j]; //brikkenavn fra "pieceFasit".

if (string1 != string2) //Hvis de er ulike settes "equal" til false.

{

equal = false;

}

}

if (equal) /\* ---- Du vant! ---- \*/

{

gamestate = 3; //Spillstatusen settes til ferdigmodus.

EventhandlerState(false); //Eventhandlerene til brikkene fjernes slik at de ikke kan klikkes på.

index = Array.IndexOf(pieceNames, "empty"); //Finner den manglende biten.

Point missingpiece = pieceCoordinates[index]; //Finner koordinatene til den manglende biten.

PictureBox newPiece = new PictureBox(); //Lager en ny brikke.

newPiece.Width = pieceSize; //Bredde.

newPiece.Height = pieceSize; //Høyde.

newPiece.Location = NameToRealCoordinates("empty"); //plasseringen til den siste brikken angis.

Bitmap imagePart = CropImage(piecesPerSide, usedPicture, missingpiece.X, missingpiece.Y); //Metoden returnerer et beskjært bilde.

newPiece.Image = imagePart; //Det beskjærte bildet settes inn i imageboxen.

newPiece.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage; //SizeMode skal være StretchImage.

newPiece.Name = "empty"; //Navnet til den siste brikken angis.

this.Controls.Add(newPiece); //legger til brikken i spillet.

tmrSec.Stop(); //Klokken i spillet stoppes.

btnStart.BackColor = Color.LightGray; //Hovedknappen blir grå.

btnStart.Text = "Lukk vindu"; //Hovedknappen endrer tekst.

MessageBox.Show("Du vant! \r\rTid:\t\t" + clock.ToString() + "\rAntall trekk:\t" + counter); //Det kommer opp en "Du Vant"-tekstboks med linjeskift og tabulatortegn.

}

}

void UpdatePositions(int steps) //Metoden flytter brikkene til der de skal være. "steps" angir hvor mange steg brikkene skal flyttes i.

{

string name;

int remainingX = 0;

int remainingY = 0;

for (int i = steps; i > 0; i--) //Flytter brikkene "steps" antall ganger.

{

foreach (Control ctrl in this.Controls) //Går gjennom alt.

{

if ((ctrl is PictureBox)) //Finner brikkene

{

name = ((PictureBox)ctrl).Name; //Finner brikkenavnet.

Point targetPosition = NameToRealCoordinates(name); //Målposisjonen.

remainingX = targetPosition.X - ((PictureBox)ctrl).Location.X; //finner gjenstående avstand langs x-aksen.

remainingY = targetPosition.Y - ((PictureBox)ctrl).Location.Y; //finner gjenstående avstand langs y-aksen.

if (remainingX != 0 || remainingY != 0) //Hvis ikke brikken er på riktig plass, skal den få ny posisjon.

{

((PictureBox)ctrl).Location = new Point(((PictureBox)ctrl).Location.X + (remainingX / i), ((PictureBox)ctrl).Location.Y + (remainingY / i)); //Brikkene flyttes 1/i (en i-del), slik at de flytter seg like mye for hver gang. De flytter seg først 1/20, så 1/19, så 1/18 av den gjenstående avstanden osv.

}

}

}

Application.DoEvents(); //Sørger for at spillet ikke stopper helt opp, selv om det venter. Andre deler av koden kan kjøres.

System.Threading.Thread.Sleep(6); //Vent i 6 millisekunder.

}

}

void RandomPositionsClickByClick(int repeats) //Metoden blander brikkene ved å "trykke på dem" repeats antall ganger. Det er bare arrayet som blir endret, ikke brikkene.

{

int choosePiece;

int index;

Point MissingButton;

Point randomClick;

for (int i = 0; i < repeats; i++) //Repeats er antall ganger det "trykkes" på en tilfeldig knappp.

{

choosePiece = r.Next(0, (piecesPerSide \* 2)-2); //Det velges et tilfeldig tall. Antall forskjellige randomverdier = antall knapper det er lov å trykke på.

index = Array.IndexOf(pieceNames, "empty");

MissingButton = pieceCoordinates[index]; //Finner koordinatene til den manglende brikken.

if (choosePiece < piecesPerSide-1) //Tallene deles i to: Nederste halvdel gjelder for trykk ved siden av tomrommet. Øverste halvdel gjelder for trykk over/under tomrommet.

{

if (choosePiece >= MissingButton.X)

{

choosePiece++; //det plusses på en for å unngå at det trykkes på tomrommet.

}

randomClick = new Point(choosePiece, MissingButton.Y); //RandomClick får tilfeldige koordinater(blant brikkene som er lov å trykke på).

}

else

{

choosePiece -= (piecesPerSide - 1);

if (choosePiece >= MissingButton.Y)

{

choosePiece++; //det plusses på en for å unngå at det trykkes på tomrommet.

}

randomClick = new Point(MissingButton.X, choosePiece); //RandomClick får tilfeldige koordinater(blant brikkene som er lov å trykke på).

}

RearrangePieces(randomClick); //Til sutt kjøres metoden som bytter om på brikkene i arrayet.

}

}

void RearrangePieces(Point piecePressed) //Metoden bytter om rekkefølgen på brikkene i arrayet "pieceNames". Piecepressed er knappen som ble trykket.

{

int indexPaste = Array.IndexOf(pieceNames, "empty"); //Finner posisjonen i arrayet til den manglende brikken.

int indexCut = 0;

int locationDifference = 0;

int upOrDown;

Point noPiece = pieceCoordinates[indexPaste]; //Finner posisjonen til tomrommet.

if (noPiece.X == piecePressed.X) //Hvis knappen som ble trykket på og tomrommet er ved siden av hverandre:

{

locationDifference = piecePressed.Y - noPiece.Y; //Finner avstanden mellom tomrommet og brikken som ble klikket.

if (locationDifference > 0) //Velger hvilken retning brikken skal flyttes i.

{

upOrDown = 1;

}

else

{

upOrDown = -1;

}

for (int i = 0; !(i == locationDifference); i += upOrDown) //gir ny posisjon til en og en brikke.

{

indexCut = Array.IndexOf(pieceCoordinates, new Point(noPiece.X, noPiece.Y + i + upOrDown)); //Søk etter brikken som er "i" brikker unna mellomrommet. Hver runde av for loopen tar en ny brikke.

string pieceToMove = pieceNames[indexCut]; //Lagre navnet til brikken som skal flyttes i en variabel.

pieceNames[indexPaste] = pieceToMove; //Lim inn navnet på den flyttede brikken i sin nye posisjon i arrayet.

pieceNames[indexCut] = "empty"; //Skriv "empty" på posisjonen i Arrayet der brikken ble fjernet fra.

indexPaste = indexCut; //Denne linjen er nødvendig fordi "empty har flyttet seg ett hakk, og da må neste runde av for-loopen vite det.

}

}

if (noPiece.Y == piecePressed.Y) //Hvis knappen som ble trykket på og tomrommet er over hverandre:

{

locationDifference = piecePressed.X - noPiece.X; //Finner avstanden mellom tomrommet og brikken som ble klikket.

if (locationDifference > 0) //Velger hvilken retning brikken skal flyttes i.

{

upOrDown = 1;

}

else

{

upOrDown = -1;

}

for (int i = 0; !(i == locationDifference); i += upOrDown) //Gir ny posisjon til en og en brikke.

{

indexCut = Array.IndexOf(pieceCoordinates, new Point(noPiece.X + i + upOrDown, noPiece.Y)); //Søk etter brikken som er "i" brikker unna mellomrommet. Hver runde av for loopen tar en ny brikke.

string pieceToMove = pieceNames[indexCut]; //Lagre navnet til brikken som skal flyttes i en variabel

pieceNames[indexPaste] = pieceToMove; //Lim inn navnet på den flyttede brikken i sin nye posisjon i arrayet

pieceNames[indexCut] = "empty"; //Skriv "empty" på posisjonen i Arrayet der brikken ble fjernet fra.

indexPaste = indexCut; //Denne linjen er nødvendig fordi "empty har flyttet seg ett hakk, og da må neste runde av for-loopen vite det.

}

}

}

Point NameToRealCoordinates(string pieceName) //Metoden bruker navnet til brikken og returnerer koordinatene brikken skal ha.

{

int index = Array.IndexOf(pieceNames, pieceName); //Finner posisjonen i arrayet.

int xoffset = (ClientRectangle.Width - pieceSize \* piecesPerSide) / 2; //mellomrommet på venstresiden av spillet.

Point position = pieceCoordinates[index];

position = new Point(position.X \* pieceSize + xoffset, position.Y \* pieceSize + topSpace); //Regner ut posisjonen brikken skal ha.

return position; //returnerer koordinatene

}

void CalculateSizeAndPositions() //Metoden regner ut avstander og størrelser til knapper og texstbokser.

{

topSpace = ClientRectangle.Height / 20 + 80; //topspace = mellomrommet over spillet.

int formWidth = ClientRectangle.Width; //Bredden på vinduet.

int formHeight = ClientRectangle.Height; //Høyden på vinduet.

int scaleFactor; //faktor for skalering av størrelsen på spillet.

int halfWidth = ClientRectangle.Width / 2; //Halve bredden av vinduet.

int xoffset;

if (formHeight - topSpace < formWidth) //Velger om det er høyden eller bredden på vinduet som skal bestemme "scalefactor".

{

scaleFactor = formHeight - topSpace;

}

else

{

scaleFactor = formWidth;

}

pieceSize = scaleFactor / piecesPerSide; //Regner ut størrelsen på hver enkelt brikke

xoffset = (ClientRectangle.Width - scaleFactor) / 2; //Mellomrommet på venstresiden av spillet.

btnStart.Width = scaleFactor / 3; //Bredden på btnStart justerer seg ved resize av spillet.

btnStart.Location = new Point(halfWidth - btnStart.Width/2,40); //btnStart er alltid midt i vinduet (horisontalt).

txtTime.Location = new Point(halfWidth - txtTime.Width / 2, 5); //Tekstboksen er alltid midt i vinduet (horisontalt).

btnMix.Location = new Point(xoffset + scaleFactor \* 5/6 -btnMix.Width/2, 40); //btnMix er plasset 5/6 av spillbredden, fra venstrekanten av spillet.

txtCounter.Location = new Point(xoffset + scaleFactor \* 1 / 6 - txtCounter.Width / 2, 64); //txtCounter er plasset 1/6 av spillbredden, fra venstrekanten av spillet.

txtCounter2.Location = new Point(xoffset + scaleFactor \* 1 / 6 - txtCounter2.Width / 2, 41); //txtCounter1 er plasset 1/6 av spillbredden, fra venstrekanten av spillet.

txtTime.Text = clock.ToString(); //Clock er en timespan variabel som konverteres til string, og legges inn en tekstboks. Da vises teksten i klokkeformat.

}

private void game\_SizeChanged(object sender, EventArgs e) //Event som kjører når størrelsen på vinduet endres.

{

string name = "";

CalculateSizeAndPositions(); //Det kjøres en metode som oppdaterer posisjon og størrelse til elementer i vinduet.

foreach (Control ctrl in this.Controls) //Går gjennom hver "Control".

{

if (ctrl is PictureBox) //Hvis det aktuelle elementet er av typen PictureBox, skal det få ny posisjon og størrelse.

{

// Sett ny posisjon

name = ((PictureBox)ctrl).Name; // bruker Casting for å behandle ctrl som en Picturebox.

((PictureBox)ctrl).Location = NameToRealCoordinates(name);

//Sett ny størelse

((PictureBox)ctrl).Width = pieceSize;

((PictureBox)ctrl).Height = pieceSize;

}

}

}

private void tmrSec\_Tick(object sender, EventArgs e) //Tick-event som skjer hvert sekund.

{

clock = TimeSpan.FromSeconds(clock.TotalSeconds + 1); //Tiden gjøres om til sekunder, og det plusses på et sekund. Alt legges tilbake i timespan variablen.

txtTime.Text = clock.ToString(); //Converterer Clock til string, og legger det inn i en tekstboks.

}

void EventhandlerState(bool state) // Fjerner eller legger til eventhandlere for alle pictureboxer. Hvis "state" = true, skal eventhandlerene legges til, med mindre de er der fra før.

{

if (eventhandlerstate != state) //Det skal ikke gjøres noe hvis eventhandlerene allerede er slik de skal være.

{

if (state) //Hvis de skal legges til:

{

foreach (Control ctrl in this.Controls) //Gå gjennom alt

{

if (ctrl is PictureBox) //Gjør dette med alle pictureboxer:

{

ctrl.Click += new System.EventHandler(this.gameButton\_Click); //Legg til en ny eventhandler

}

}

}

else //Hvis eventhandlerne skal fjernes:

{

foreach (Control ctrl in this.Controls) //Gå gjennom alt

{

if (ctrl is PictureBox) //Gjør dette med alle pictureboxer:

{

ctrl.Click -= new System.EventHandler(this.gameButton\_Click); //Fjern eventhandleren

}

}

}

eventhandlerstate = !eventhandlerstate; // Inverter verdien i variabelen eventhandlerstate. Den holder styr på om eventhandlerene er på eller av

}

}

private void btnStart\_Click(object sender, EventArgs e) //Eventhandler for btnStart-Knappen

{

switch (gamestate) // Gamestate variabelen forteller om spillet er ustartet, startet, om det er i pause eller om det er feridg.

{

case 0: // Ikke startet --> spillet startes

btnStart.BackColor = Color.Gold; //Endrer farge på btnStart-Knappen

gamestate = 1; //Endrer gamestate

tmrSec.Start(); //Starter timeren

EventhandlerState(true); //Slår på Eventhandlerene

btnStart.Text = "Pause"; //Endrer teksten på btnStart-Knappen til "pause" fordi spillet settes i gang, og neste trykk på knappen vil sette spillet i pause.

RandomPositionsClickByClick(50); //Blander brikkene 50 ganger

UpdatePositions(pieceSize / 16); // Oppdaterer posisjonene til hver enkelt brikke etter at de er blandet.

break; // Break indikerer at "case 0" er ferdig

case 1: // Startet --> spillet stoppes

btnStart.Text = "Fortsett"; // Teksten på btnStart-Knappen settes til "fortsett"

btnStart.BackColor = Color.LimeGreen; // btnStart-Knappen blir grønn

tmrSec.Stop(); //Stopper timeren.

EventhandlerState(false); //Fjerner eventhandlerne, slik at det ikke er mulig å flytte på brikkene.

gamestate = 2; //Endrer gamestate

break;

case 2: //Stoppet --> spillet startes

btnStart.Text = "Pause"; // btnStart-Knappen får teksten pause

btnStart.BackColor = Color.Gold; //btnStart - Knappen får gullfarge

tmrSec.Start(); //Timeren startes

EventhandlerState(true); //Eventhandlerne kommer tilbake

gamestate = 1; //Gamestate endres

break;

case 3: // Spillet er feridg

this.Close(); //Vinduet lukkes

break;

}

}

private void btnMix\_Click(object sender, EventArgs e) //Mix-knappen blander knappene ved å kjøre to metoder.

{

RandomPositionsClickByClick(50); //Metoden blander arrayet som holder styr på posisjonene til brikkene

UpdatePositions(pieceSize / 16); //Metoden flytter brikkene til der de skal være.

}

//\* ------------------- Metode inspirert av Arne Wallerud ------------------- \*/

Bitmap CropImage(int partsPerSide, Bitmap original, int x, int y) //Metoden returnerer et bitmap bilde som er en beskjært versjon av originalbildet. Metoden er basert på deler fra Arne Wallerud sitt programm.

{

// partsPerSide forteller hvor mange biter bildet skal deles opp i horisontalt og vertikalt. partsPerSide = 3 vil si at bildet deles opp i 9.

// variablene x og y er koordinater som forteller hvilken av del av bildet som skal returneres.

int fullImageWidth = original.Width; //Lagrer bredden på bildet i en variabel

int fullImageHeight = original.Height; //Lagrer høyden på bildet i en variabel

int imageWidth = fullImageWidth / partsPerSide; //Bredden på det beskjærte bildet skal være totalbredden delt på antall brikker i bredden.

int imageHeight = fullImageHeight / partsPerSide; //Høyden på det beskjærte bildet skal være totalhøyden delt på antall brikker i høyden.

Rectangle partOfImage = new Rectangle(imageWidth \* x, imageHeight \* y, imageWidth, imageHeight); //Rektangelet er den delen av bildet som skal klippes ut. Det lages et rektangel der hjørnet øverst til venstre bestemmes av x og y koordinatene ganget med avstanden mellom brikkene. høyden og bredden på rektangelet bestemmes av imageWidth og imageHeight.

Bitmap cropped = (Bitmap)original.Clone(partOfImage, original.PixelFormat); //Clone-metoden til bildet blir brukt, og det lages et nytt bilde (castet til bitmap) der rektangelet som er laget brukes til å bestemme hvilken del av bildet som skal brukes.

return cropped; // Det beskjærte bildet returneres.

}

}

}