|  |  |
| --- | --- |
|  | class vehicle: |
|  |  |
|  |  |
|  | def updatePosition(self,recto): |
|  | self.rect=recto |
|  | x=int((2\*self.rect[0]+self.rect[2])/2) |
|  | y=int((2\*self.rect[1]+self.rect[3])/2) |
|  | self.points.append((x,y)) |
|  | self.diagonal=(self.rect[2]\*\*2+self.rect[3]\*\*2)\*\*0.5 |
|  |  |
|  | def predictNext(self): |
|  | if len(self.points)==1: |
|  | x=self.points[0][0] |
|  | y=self.points[0][1] |
|  | self.next=[x,y] |
|  | elif len(self.points)==2: |
|  | delx=self.points[1][0]-self.points[0][0] |
|  | dely=self.points[1][1]-self.points[0][1] |
|  | self.next=[self.points[1][0]+delx,self.points[1][1]+dely] |
|  | elif len(self.points)==3: |
|  | delx=((self.points[2][0]-self.points[1][0])\*2+(self.points[1][0]-self.points[0][0]))/3 |
|  | dely=((self.points[2][1]-self.points[1][1])\*2+(self.points[1][1]-self.points[0][1]))/3 |
|  | self.next=[int(self.points[2][0]+delx),int(self.points[2][1]+dely)] |
|  | elif len(self.points)==4: |
|  | delx=((self.points[3][0]-self.points[2][0])\*3+(self.points[2][0]-self.points[1][0])\*2+(self.points[1][0]-self.points[0][0]))/6 |
|  | dely=((self.points[3][1]-self.points[2][1])\*3+(self.points[2][1]-self.points[1][1])\*2+(self.points[1][1]-self.points[0][1]))/6 |
|  | self.next=[int(self.points[3][0]+delx),int(self.points[3][1]+dely)] |
|  | elif len(self.points)>=5: |
|  | delx=((self.points[-1][0]-self.points[-2][0])\*4+(self.points[-2][0]-self.points[-3][0])\*3+(self.points[-3][0]-self.points[-4][0])\*2+(self.points[-4][0]-self.points[-5][0]))/10 |
|  | dely=((self.points[-1][1]-self.points[-2][1])\*4+(self.points[-2][1]-self.points[-3][1])\*3+(self.points[-3][1]-self.points[-4][1])\*2+(self.points[-4][1]-self.points[-5][1]))/10 |
|  | self.next=[int(self.points[-1][0]+delx),int(self.points[-1][1]+dely)] |
|  |  |
|  |  |
|  | def updatePosition(self,recto): |
|  | self.rect=recto |
|  | x=int((2\*self.rect[0]+self.rect[2])/2) |
|  | y=int((2\*self.rect[1]+self.rect[3])/2) |
|  | self.points.append((x,y)) |
|  | self.diagonal=(self.rect[2]\*\*2+self.rect[3]\*\*2)\*\*0.5 |
|  |  |
|  | def increaseFrameNotFound(self): |
|  | self.framesNotFound+=1 |
|  | if(self.framesNotFound>5): |
|  | self.tracking=False |
|  |  |
|  | def setCurrentFrameMatch(self,bool): |
|  | self.currentframeMatch=bool |
|  |  |
|  | def getCurrentFrameMatch(self): |
|  | return self.currentframeMatch |
|  |  |
|  | def getTracking(self): |
|  | return self.tracking |
|  |  |
|  | def getNext(self): |
|  | return self.next |
|  |  |
|  | def getDiagonal(self): |
|  | return self.diagonal |
|  |  |
|  | def getPoints(self): |
|  | return self.points |
|  |  |
|  | def getRectangle(self): |
|  | return self.rect |
|  |  |
|  |  |
|  |  |
|  |  |
|  | def \_\_init\_\_(self,rect=[]): |
|  | self.points=[] |
|  | self.rect=rect |
|  | self.crossed=False |
|  | self.tracking=True |
|  | self.speedChecked=False |
|  | self.entered=False |
|  | self.exited=False |
|  | self.enterTime=0.0 |
|  | self.exitTime=0.0 |
|  | self.currentframeMatch=True |
|  | self.framesNotFound=0 |
|  | x=int((2\*self.rect[0]+self.rect[2])/2) |
|  | y=int((2\*self.rect[1]+self.rect[3])/2) |
|  | self.points.append((x,y)) |
|  | self.next=[x,y] |
|  | self.diagonal=(self.rect[2]\*\*2+self.rect[3]\*\*2)\*\*0.5 |